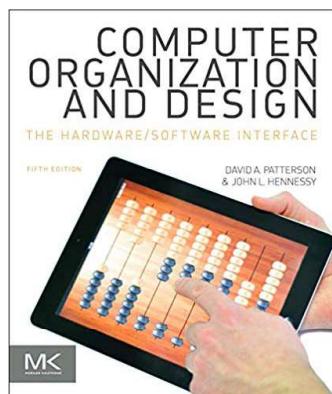


# Aula 11 - Pipeline Avançado

Prof. Dr. Clodoaldo A. de Moraes Lima

Material baseado no livro “Patterson, David A., Hennessy, J. L. - Computer Organization And Design: The Hardware/Software Interface”



## Variantes do Pipeline já analisadas

- Único ciclo

- Todas as instruções são executadas num só ciclo de relógio
- Pouco eficiente, nunca utilizada em processadores comerciais.
- CPI = 1
- $T_{cc}$  = duração da instrução mais longa
- Latência de uma instrução =  $T_{cc}$

- Multi-ciclo

- Cada instrução demora vários ciclos a executar.
- CPI = varia de 1 até CPImax (número máximo de ciclos numa instrução - depende do mix de instruções)
- $T_{cc}$  = duração de cada ciclo
- Latência =  $T_{cc} \times 1 \dots CPI_{max}$

## Variantes do Pipeline já analisadas

- Pipeline

- cada fase (estágio) da pipeline executa de forma concorrente, como numa linha de montagem, melhorando o CPI relativamente à implementação multi-ciclo
- CPI = 1 (**em condições ideais**)
- Tcc = duração do estágio mais longo da pipeline
- Latência = Tcc x número de estágios da pipeline

T	SEM Pipeline					COM Pipeline						
	BI	DI	CO	BO	EI	EO	BI	DI	CO	BO	EI	EO
0	I1	-	-	-	-	-	I1	-	-	-	-	-
1	-	I1	-	-	-	-	I2	I1	-	-	-	-
2	-	-	I1	-	-	-	I3	I2	I1	-	-	-
3	-	-	-	I1	-	-	I4	I3	I2	I1	-	-
4	-	-	-	-	I1	-	I5	I4	I3	I2	I1	-
5	-	-	-	-	-	I1	I6	I5	I4	I3	I2	I1
6	I2	-	-	-	-	-	I7	I6	I5	I4	I3	I2
7	-	I2	-	-	-	-	I8	I7	I6	I5	I4	I3
8	-	-	I2	-	-	-	I9	I8	I7	I6	I5	I4
9	-	-	-	I2	-	-	I10	I9	I8	I7	I6	I5
10	-	-	-	-	I2	-	I11	I10	I9	I8	I7	I6
11	-	-	-	-	-	I2	I12	I11	I10	I9	I8	I7
12	I3	-	-	-	-	-	I13	I12	I11	I10	I9	I8

## Variantes do Pipeline já analisadas

- O desempenho de uma dada arquitetura é dado pela combinação dos vários fatores:

$$T_{exec} = NINST \times CPI \times Tcc$$

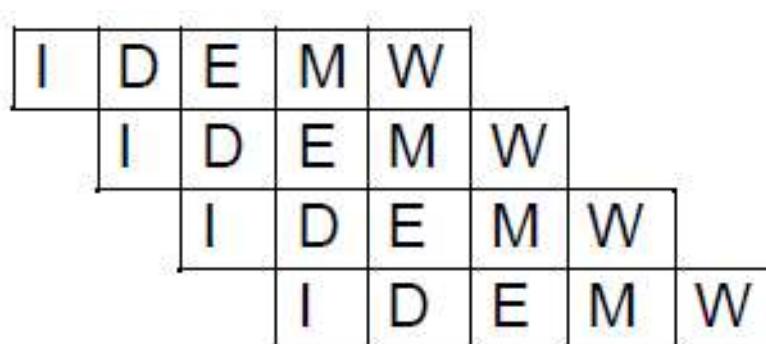
- A latência da execução das instruções é um fator importante, principalmente no caso dos saltos, porque influênciaria o número de ciclos que é necessário carregar a pipeline

# Pipeline Avançado: Resumo

## Pipelining - Execução sobreposta das instruções



Super-pipelining - Aumenta a frequência interna de relógio  
(→+estágios)



# Pipeline Avançado: Resumo

VLIW (EPIC) - Especifica várias operações paralelas por instrução

IF	ID	EXE	MEM	WB
		EXE	MEM	WB
		EXE	MEM	WB
		EXE	MEM	WB

Instruções vetoriais - especifica uma série de operações a realizar em vários dados, numa só instrução

IF	ID	EXE	MEM	WB
		EXE	MEM	WB
		EXE	MEM	WB
		EXE	MEM	WB

# Arquitetura Superpipeline

Anteriormente, foi apresentado o conceito de "pipeline", isto é, uma forma de organizar o funcionamento da CPU para que ela seja capaz de processar as instruções mais rapidamente, minimizando a ocorrência de circuitos ociosos.

Seqüência no Tempo	SEM pipeline		COM pipeline	
	Busca	Execução	Busca	Execução
0	I1	-	I1	-
1	-	I1	I2	I1
2	I2	-	I3	I2
3	-	I2	I4	I3
4	I3	-	I5	I4

# Arquitetura Superpipeline

Alguns pesquisadores, entretanto, perceberam que, quando se subdividia a pipeline em um maior número de níveis, alguns destes estágios precisavam de muito menos tempo de execução que um ciclo de clock.

T	SEM Pipeline						COM Pipeline					
	BI	DI	CO	BO	EI	EO	BI	DI	CO	BO	EI	EO
0	I1	-	-	-	-	-	I1	-	-	-	-	-
1	-	I1	-	-	-	-	I2	I1	-	-	-	-
2	-	-	I1	-	-	-	I3	I2	I1	-	-	-
3	-	-	-	I1	-	-	I4	I3	I2	I1	-	-
4	-	-	-	-	I1	-	I5	I4	I3	I2	I1	-
5	-	-	-	-	-	I1	I6	I5	I4	I3	I2	I1
6	I2	-	-	-	-	-	I7	I6	I5	I4	I3	I2
7	-	I2	-	-	-	-	I8	I7	I6	I5	I4	I3
8	-	-	I2	-	-	-	I9	I8	I7	I6	I5	I4
9	-	-	-	I2	-	-	I10	I9	I8	I7	I6	I5
10	-	-	-	-	I2	-	I11	I10	I9	I8	I7	I6
11	-	-	-	-	-	I2	I12	I11	I10	I9	I8	I7
12	I3	-	-	-	-	-	I13	I12	I11	I10	I9	I8

# Arquitetura Superpipeline

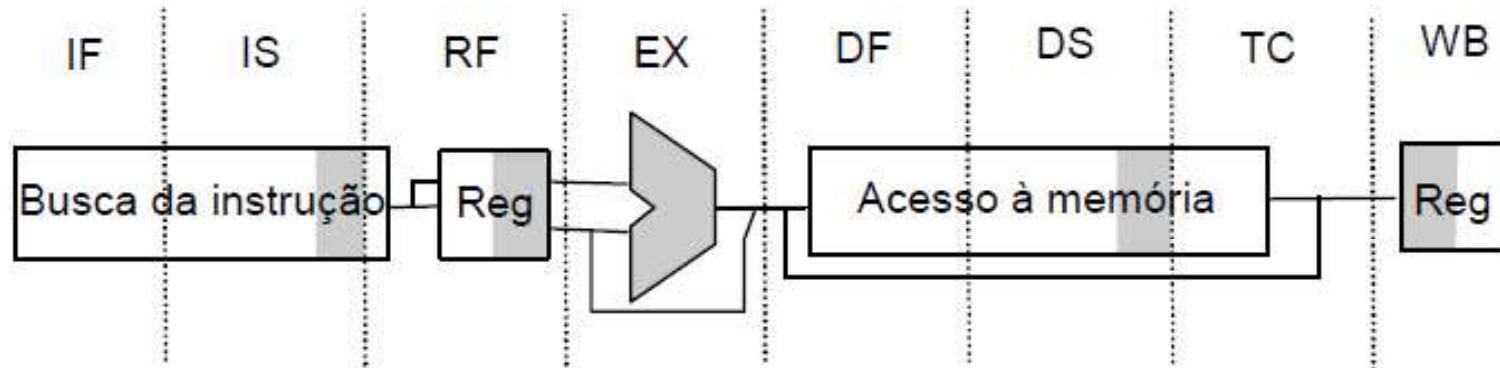
Assim, foi proposto o uso de um duplicador (ou até triplicador) de clock interno na CPU, de maneira que muitos destes estágios do pipeline pudessem ser finalizados na metade do tempo, acelerando ainda mais o processamento final.

É importante notar, porém, que o aumento de desempenho é oriundo de um menor tempo de processamento de cada instrução, mas elas ainda estão sendo executadas uma a uma.

# Super-pipelining

- Consiste em aumentar o número de estágios da pipeline, conseguindo diminuir Tcc e aumentar a frequência de relógio.
- Pode não diminuir a latência das instruções (pode mesmo aumentar) o que implica um aumento das penalizações devido a anomalias de dados e de controle, se não forem introduzidos outros melhoramentos no caminho dos dados
- MIPS R4000 – 8 estágios de pipeline, sendo os estágios adicionais utilizados para acessos à memória

# Super-pipelining



IF – Primeira metade da busca da instrução (seleção do PC)

IS – Segunda metade da busca da instrução, completando o acesso

RF – Decodificação e leitura dos valores dos registradores, detecção de hit

EX – Execução, incluindo a resolução dos saltos

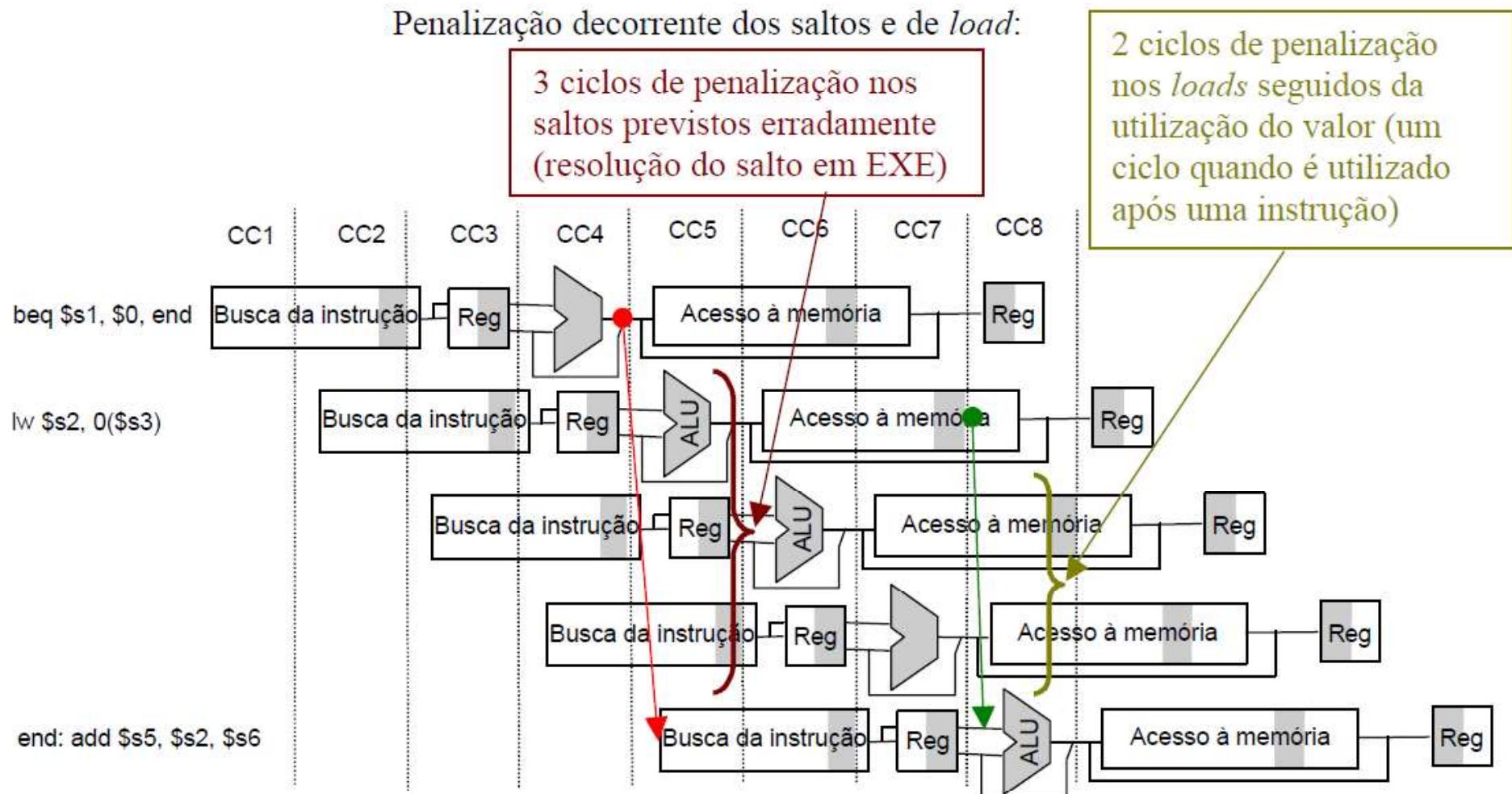
DF – Busca dos dados, primeira metade de acesso a cache

DS – Segunda metade do acesso a cache

TC – Tag check, determinar se o acesso foi um hit

WB- Escrita dos resultados em registrador para load e registrador-registrador

# Super-pipelining



# Super-pipelining

Comparação do desempenho entre o MIPS com 5 estágios (MIPS5) e o MIPS R4000 com 8 estágios de pipeline (MIPS8), assumindo que não existem caches misses.

Mistura de instruções (gcc):

Tipo de Instrução	Frequência
Tipo R	49%
<i>Load</i>	22%
<i>Store</i>	11%
<i>Branch</i>	18%

# Super-pipelining

MIPS5 (resolução dos saltos em EXE, s/ otimizar escritas em PC):

$$\text{CPIload (50\% de situações de bolha)} = 0.5 \times 1.0 + 0.5 \times 2.0 = 1.5$$

$$\text{CPIbranch (75\% previsões corretas)} = 0.75 \times 1.0 + 0.25 \times 3.0 = 1.5$$

$$\text{CP15} = 0.49 \times 1.0 + 0.22 \times 1.5 + 0.11 \times 1.0 + 0.18 \times 1.5 = 1.2$$

MIPS8

$$\text{CPIload (50\% de situações de bolha)} = 0.5 \times 1.0 + 0.5 \times 3.0 = 2.0$$

$$\text{CPIbranch (75\% previsões corretas)} = 0.75 \times 1.0 + 0.25 \times 4.0 = 1.75$$

$$\text{CP18} = 0.49 \times 1.0 + 0.22 \times 2.0 + 0.11 \times 1.0 + 0.18 \times 1.75 = 1.36$$

# Super-pipelining

Comparação mantendo a frequência de relógio

$$\frac{CPI5}{CPI8} = \frac{1.2}{1.36} = 0.88 \text{ (degradação de 12 %)}$$

Aumentado a frequência de relógio em 50% ( $Tcc5 = 1.5 \times Tcc8$ ):

$$\text{Ganho} = \frac{NINST \times CPI5 \times Tcc5}{NINST \times CPI8 \times Tcc8} = \frac{1.5 \times CPI5}{CPI8} = \frac{1.5 \times 1.2}{1.36} = 1.32x$$

Melhorando a previsão de saltos para 90% e reduzindo um ciclo no impacto das bolhas nos load:

$$CPI_{load} \text{ (50\% de situações de stall)} = 0.5 \times 1.0 + 0.5 \times 2.0 = 1.5$$

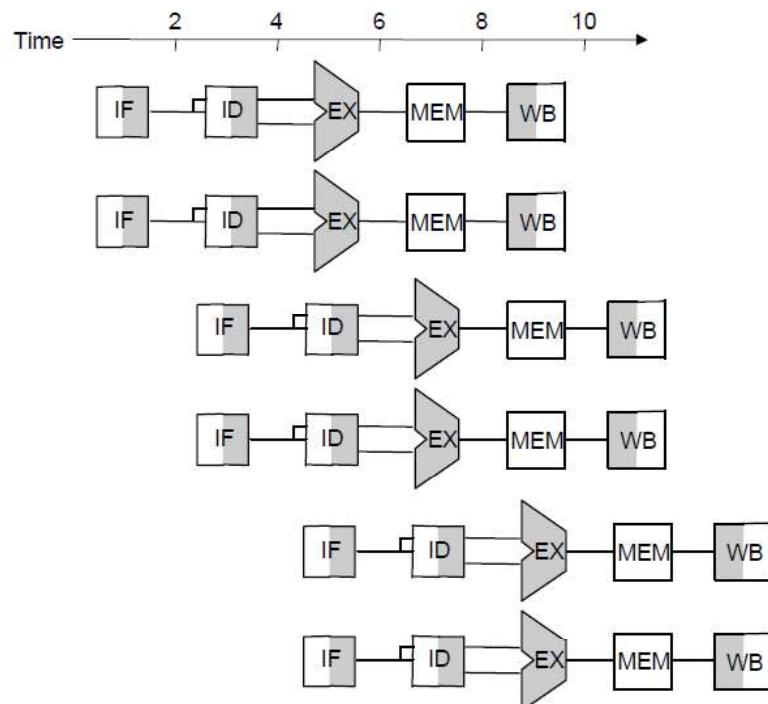
$$CPI_{branch} \text{ (90\% previsões correctas)} = 0.9 \times 1.0 + 0.1 \times 4.0 = 1.3$$

$$CPI8 = 0.49 \times 1 + 0.22 \times 1.5 + 0.11 \times 1 + 0.18 \times 1.3 = 1.16$$

$$Ganho = \frac{1.5 \times 1.2}{1.16} = 1.55x$$

# Super-escalaridade

- Baseia-se no **aumento das unidades funcionais** de forma que seja possível **executar mais de uma instrução em cada ciclo de relógio**
- $CPI < 1$
- Exemplo: MIPS com grau de super-escalaridade 2



# Super-escalaridade

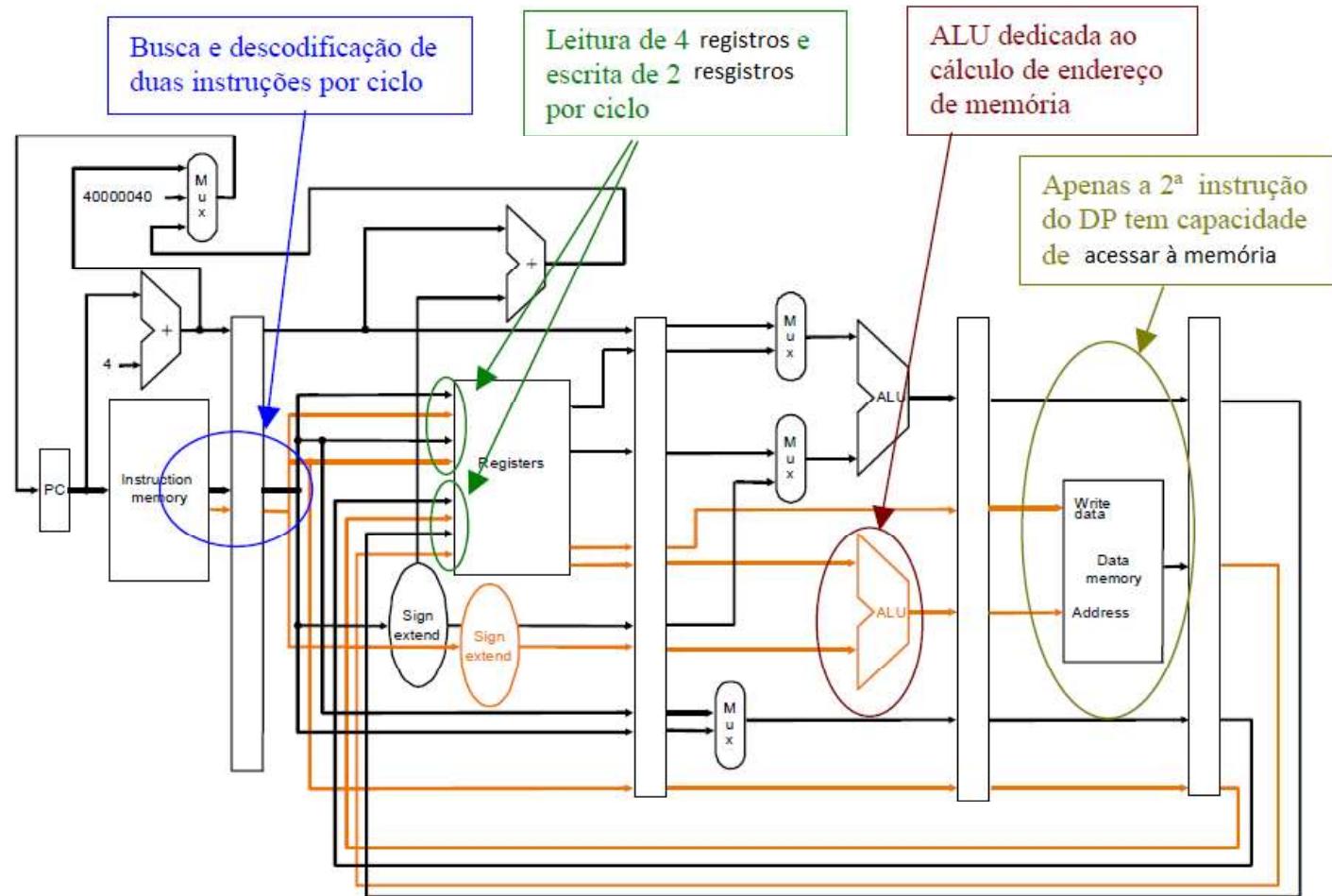
- Este tipo de abordagem requer a capacidade de efetuar a busca de duas instruções por ciclo (64 bits no MIPS), duas descodificações, etc.
- Uma abordagem mais simples consiste em apenas duplicar algumas unidades funcionais, não suportando todas as combinações de instruções.
- Em MIPS a unidade de MEM é utilizada apenas por load e store, sendo mais simples conceber um MIPS super-escalar que suporta uma operação tipo R ou branch em simultâneo com um acesso à memória.

Tipo de instrução	Estágio					
ALU ou branch	IF	ID	EX	MEM	WB	
<i>load ou store</i>	IF	ID	EX	MEM	WB	
ALU ou branch		IF	ID	EX	MEM	WB
<i>load ou store</i>		IF	ID	EX	MEM	WB
ALU ou branch			IF	ID	EX	MEM
<i>load ou store</i>			IF	ID	EX	WB

# Super-escalabilidade

## MIPS Super-escalado

- Caminho de dados suporta uma operação tipo R ou branch em simultâneo com um acesso à memória.



## MIPS Super-escalar

- Desempenho de MIPS super-escalar (ignorando penalização dos saltos).

cic:    lw      \$t0, 0(\$s1)  
        addu    \$t0, \$t0, \$s2  
        sw      \$t0, 0(\$s1)  
        addi    \$s1, \$s1, -4  
        bne     \$s1, \$0, cic

	ALU    branch	Load    store	Ciclo
cic:		lw \$t0, 0(\$s1)	1
	addi \$s1, \$s1, -4		2
	addu \$t0, \$t0, \$s2		3
	bne \$s1, \$0, cic	sw \$t0, 4(\$s1)	4

$$\text{CPI} \cong \text{número de ciclos} / \text{número de instruções} = 4 / 5 = 0,8$$

## MIPS Super-escalar

- Desempenho de MIPS super-escalar com loop unrolling (grau 4)
  - juntar várias iterações num simples ciclo
  - amortiza overhead dos ciclos em várias iterações

	ALU    branch	Load    store	Ciclo
cic:	addi \$s1, \$s1, -16	lw \$t0, 0(\$s1)	1
		lw \$t1, 12(\$s1)	2
	addu \$t0, \$t0, \$s2	lw \$t2, 8(\$s1)	3
	addu \$t1, \$t1, \$s2	lw \$t3, 4(\$s1)	4
	addu \$t2, \$t2, \$s2	sw \$t0, 0(\$s1)	5
	addu \$t3, \$t3, \$s2	sw \$t1, 12(\$s1)	6
		sw \$t2, 8(\$s1)	7
	bne \$s1, \$0, cic	sw \$t3, 4(\$s1)	8

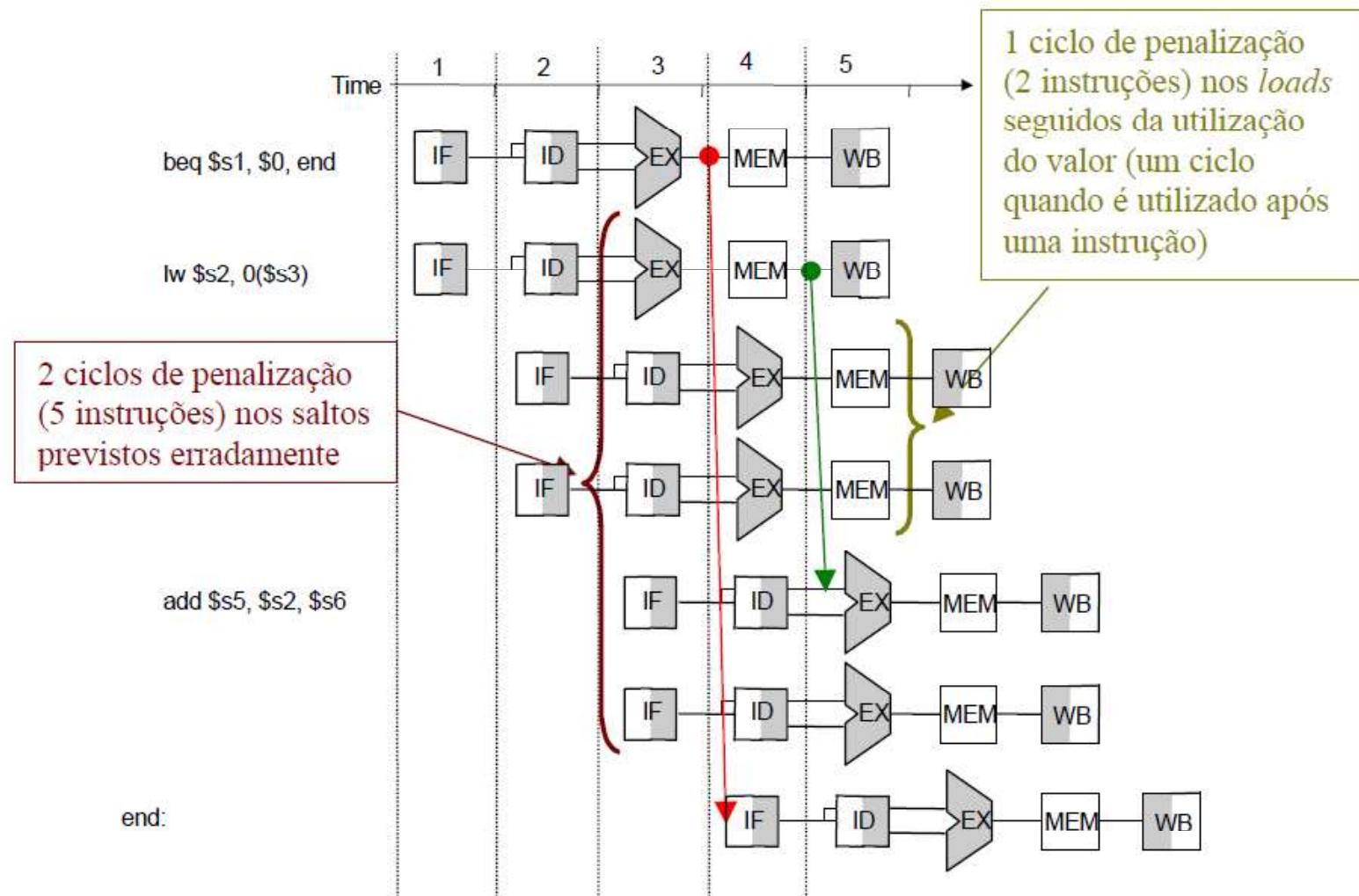
$$CPI \leq 8/14 = 0,57$$

Utiliza mais registradores  
\$t1,\$t2 e \$t3

# Super-escalaridade

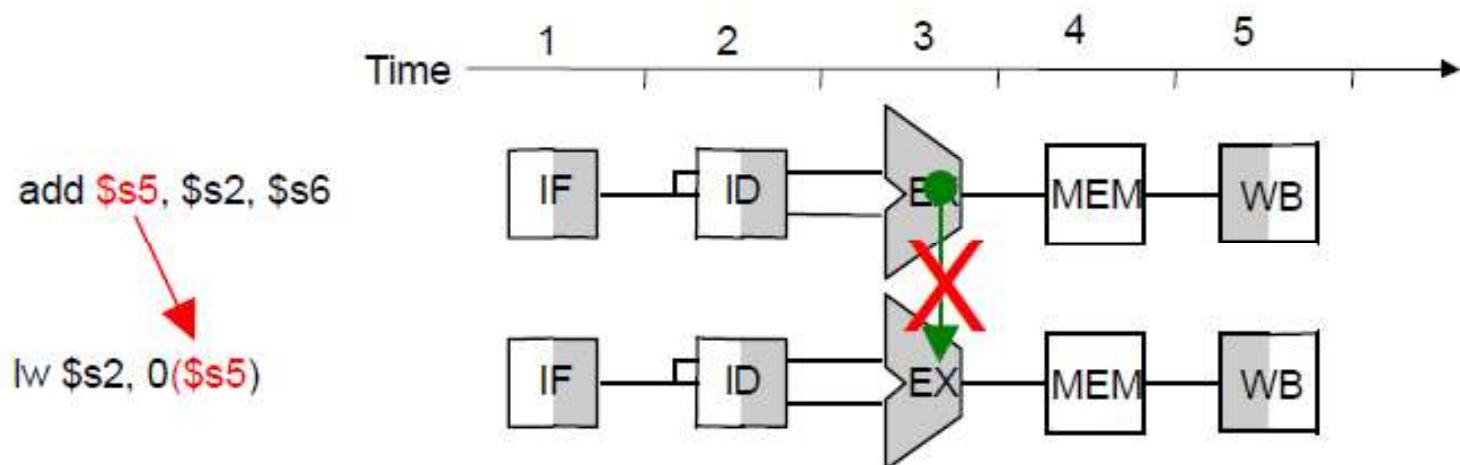
## MIPS Super-escalar

- Penalização decorrente dos saltos e de load:



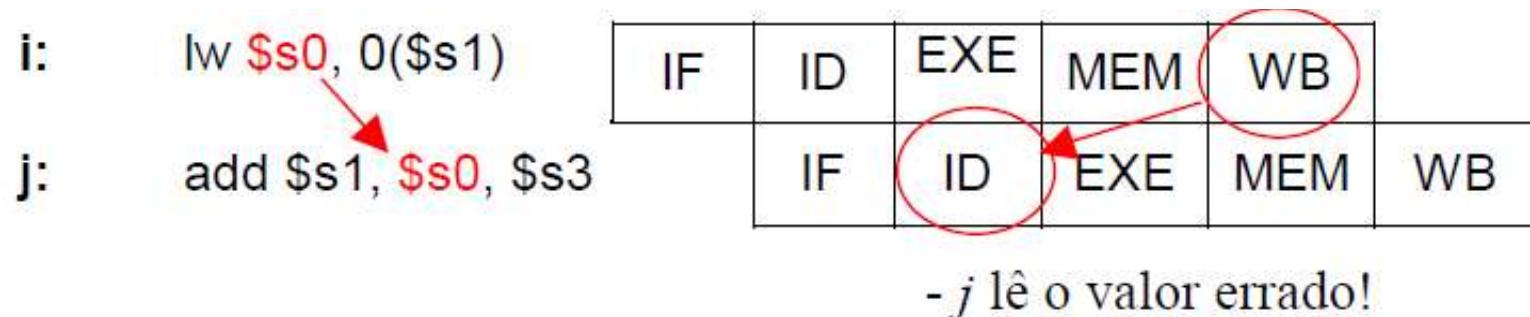
## MIPS Super-escalar

- A detecção de anomalias e encaminhamento de dados em arquiteturas super-escalares é extremamente complexa.
- Adicionalmente as instruções executadas em cada ciclo devem ser completamente independentes



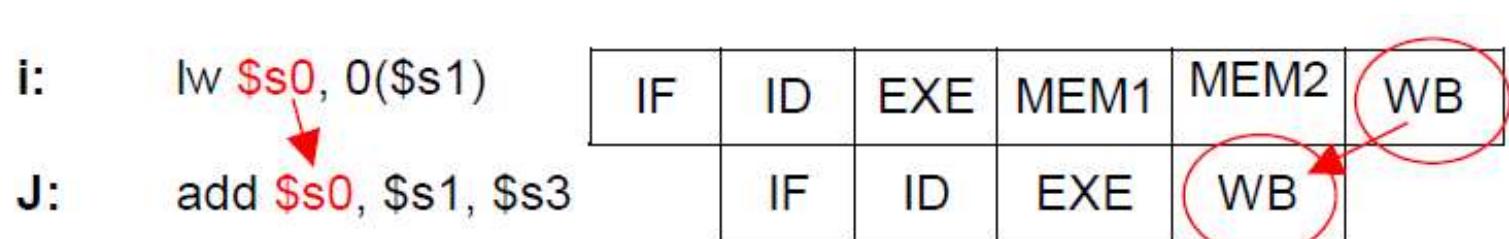
## Nova visão das dependências de dados que geram anomalias

- RAW (Read After Write) – uma instrução j tenta ler um operando antes de uma instrução anterior i o escrever.
- Normalmente resolvido com bolha ou encaminhamento de dados



## Nova visão das dependências de dados que geram anomalias

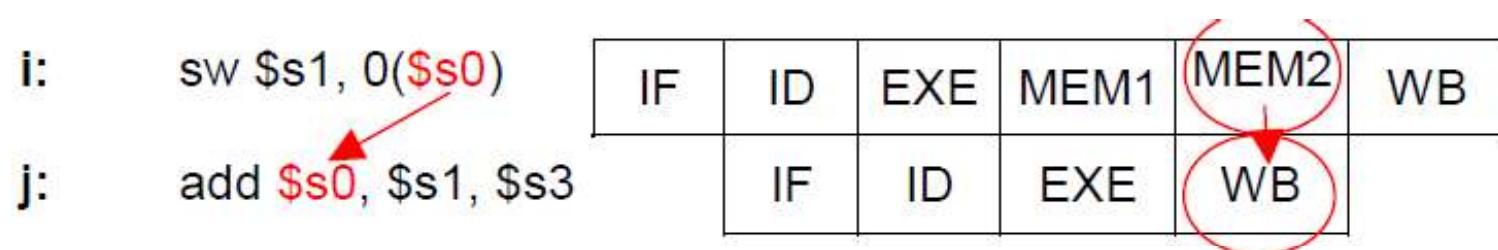
- WAW (Write After Write) – uma instrução j tenta escrever o operando antes de uma instrução anterior i o escrever.
- Está presente apenas em pipelines que escrevem valores em mais de um estágio da pipeline, ou quando as instruções não são completadas na ordem que foram geradas no programa



- o operando fica com o valor errado (instr. i)

## Nova visão das dependências de dados que geram anomalias

- WAR (Write After Read) – uma instrução j tenta escrever um novo valor num operando antes de uma instrução anterior i o ler.
- Ocorre quando a leitura dos registradores é realizada após o WB da instrução seguinte, isto é, quando as instruções não são completadas na ordem que foram geradas no programa



## Nova visão das dependências de dados que geram anomalias

- As dependências WAW e WAR resultam da utilização de um número limitado de registadores (dependências de nome), podendo ser eliminadas através da RENOMEAÇÃO de REGISTRADORES.
- As dependências de dados em acessos à memória são mais complexas:
  - $100 (\$s0) = 20 (\$s2)$  ?
  - $20 (\$s2) = 20 (\$s2)$  em iterações diferentes de um ciclo?

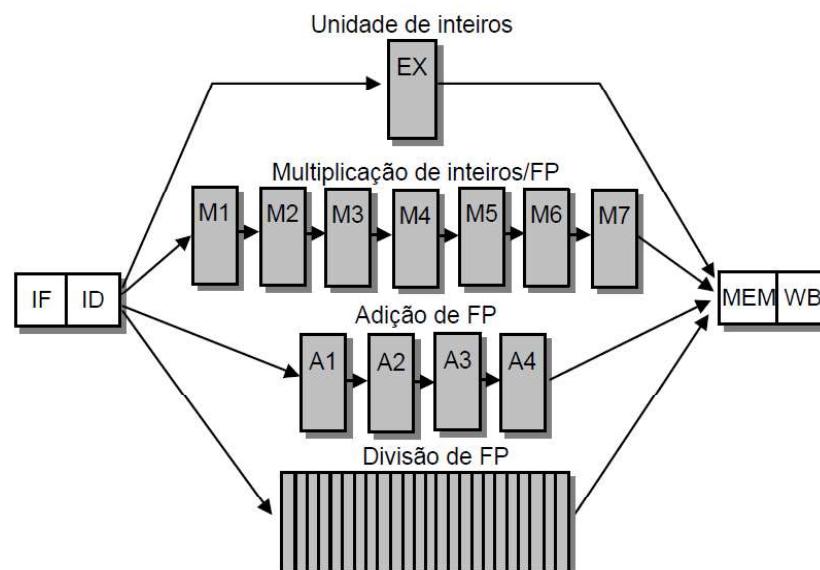
# Pipelining com operações multi-ciclo

- Na execução de um pipeline não é viável que todas as operações demorem o mesmo número de ciclos, especialmente quando são suportadas instruções com ponto flutuante .
- De forma geral as arquiteturas possuem várias unidades funcionais, cada unidade implementando uma funcionalidade específica (ex. operações em FP).
- Nem todas as unidades funcionais podem implementar a execução em pipeline. Quando é suportada pipeline o estágio EXE divide-se em vários estágios, podendo ser iniciada uma instrução em cada ciclo, caso contrário, o estágio EXE é repetido várias vezes.

# Pipelining com operações multi-ciclo

Exemplo com quatro unidades funcionais:

- ① Processamento de inteiros, processa operações na ALU em inteiros, load e store e saltos, 1 ciclo
- ② Multiplicação de Ponto Flutuante e de inteiros, 7 ciclos
- ③ Adição e subtração de Ponto Flutuante, 4 ciclos
- ④ Divisão de Ponto Flutuante e de inteiros, 24 ciclos, sem pipeline



# Pipelining com operações multi-ciclo

- A latência determina o número de ciclos que devem existir entre uma instrução que produz um valor e uma instrução que utiliza esse valor (dependências RAW, com encaminhamento de dados)
- O intervalo de iniciação determina o número de ciclos que deve separar duas instruções que utilizam a mesma unidade (para evitar anomalias estruturais)
- Exemplo usando MIPS com 4 unidades funcionais

Unidade funcional	Latência	Intervalo de iniciação
ALU de inteiros	0	1
Memória de dados (lw)	1	1
Adição FP	3	1
Multiplicação FP	6	1
Divisão FP	24	24

# Pipelining com operações multi-ciclo

- A unidade de divisão pode originar anomalias estruturais, uma vez que não implementa pipeline, o que origina bolhas (stalls)
- O número de escritas em registradores num ciclo pode ser superior a 1 (anomalia estrutural) porque a conclusão das instruções (WB) não é efetuada pela ordem que são iniciadas



# Pipelining com operações multi-ciclo

Instrução	Ciclo										
	1	2	3	4	5	6	7	8	9	10	11
MULTD F0, F4, F6	IF	ID	M1	M2	M3	M4	M5	M6	M7	MEM	WB
...	IF	ID	EXE	MEM	WB						
...	IF	ID	EXE	MEM	WB						
ADDD F2, F4, F6		IF	ID	A1	A2	A3	A5	MEM	WB		
		IF	ID	EXE	MEM	WB					
		IF	ID	EXE	MEM	WB					
LD F8, 0 (R2)			IF	ID	EXE	MEM	WB				

# Pipelining com operações multi-ciclo

- As dependências de dados (RAW) impõem limitações às instruções que podem iniciar a execução em cada ciclo (devem ser respeitadas as latência e os intervalos de iniciação) o que origina stalls mais frequentes:

Instrução	1	2	3	4	5	6	C	i	c	I	o	12	13	14	15	16
LD F4, 0(R2)	IF	ID	EX	M	WB											
MULTD F0, F4, F6		IF	ID	st	M1	M2	M3	M4	M5	M6	M7	M	WB			
ADDD F2, F0, F8			IF	st	ID	st	st	st	st	st	A1	A2	A3	A4	M	
SD F2, 0(R2)				st	IF	st	st	st	st	st	st	ID	st	st	EX	

# Pipelining com operações multi-ciclo

- Podem surgir problemas com WAW porque a conclusão das instruções (WB) não é efetuada pela ordem que são iniciadas
- As dependências WAR não causam problemas porque todas as instruções lêem os operandos (2o estágio) antes da seguinte os escrever (5o estágio ou mais tarde)
- Existem problemas com o processamento de exceções, uma vez que as instruções não são completadas pela ordem do programa

# Escalonamento

Até agora admitiu-se que as instruções são adquiridas pela ordem com que foram escritas no programa, e executadas pela mesma ordem

Contudo, sem perder o correto fluxo de dados (dataflow), são possíveis duas outras hipóteses

- O compilador pode trocar a ordem das instruções e apresentá-las ao datapath por uma ordem diferente da qual foi utilizada pelo programador, e o datapath executará então as instruções por essa ordem → **Escalonamento estático**
- Independentemente da ordem na qual as instruções chegam ao datapath, esta pode executá-las por uma ordem diferente desde que não quebre o fluxo de dados → **Escalonamento dinâmico.**

# Escalonamento

No caso em que não há escalonamento dinâmico, as instruções são emitidas (Issued) pela ordem com que são geradas pelo compilador, executadas pela mesma ordem e terminadas ainda em ordem

- Emissão em ordem, execução em ordem e termino em ordem
- Execução fora de ordem → termino fora de ordem?

Quando há escalonamento dinâmico podemos ter essencialmente duas situações

- Emissão em ordem, execução fora de ordem e termino fora de ordem
- Emissão fora de ordem, execução fora de ordem e termino fora de ordem

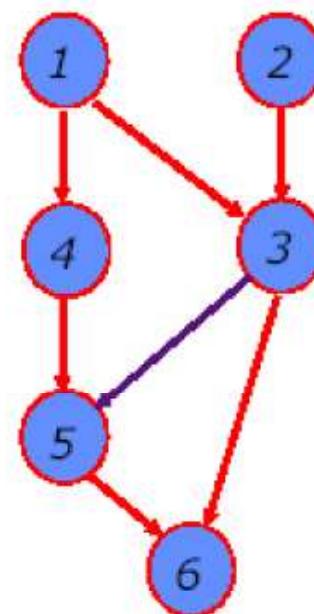
Vamos estudar as primeiras

- Na fase ID verifica se a instrução pode iniciar a fase de execução, caso contrário é originado um "stall" das instruções em IF e ID:
  - ① Não existem anomalias funcionais (apenas uma instrução pode fazer WB em cada ciclo e/ou a unidade funcional deve estar livre)
  - ② As dependências RAW podem ser resolvidas com encaminhamento (não existem instruções pendentes que escrevam num registrador utilizado)
  - ③ A instrução não origina anomalias WAW (nenhuma das instruções pendentes escreve no mesmo registrador que a atual)
- Origina entre 0,65 e 1,21 bolhas por instrução (SPEC FP)
- O escalonamento estático com a **iniciação das instruções pela ordem do programa**, quando não existem dependências é bastante limitado!

# Limitações da emissão por ordem

Pelo fato da emissão de instruções se fazerem por ordem, a instrução 4 não pode ser emitida aqui.

1	LD	F2,	34(R2)	<i>latency</i> 1
2	LD	F4,	45(R3)	<i>long</i>
3	MULTD	F6,	F4, F2	3
4	SUBD	F8,	F2, F2	1
5	DIVD	F4,	F2, F8	4
6	ADDD	F10,	F6, F4	1



# Escalonamento dinâmico

Emissão em ordem, execução fora de ordem e termino fora de ordem

## Idéia Chave

Permite que as instruções que se seguem a um "stall" possam prosseguir

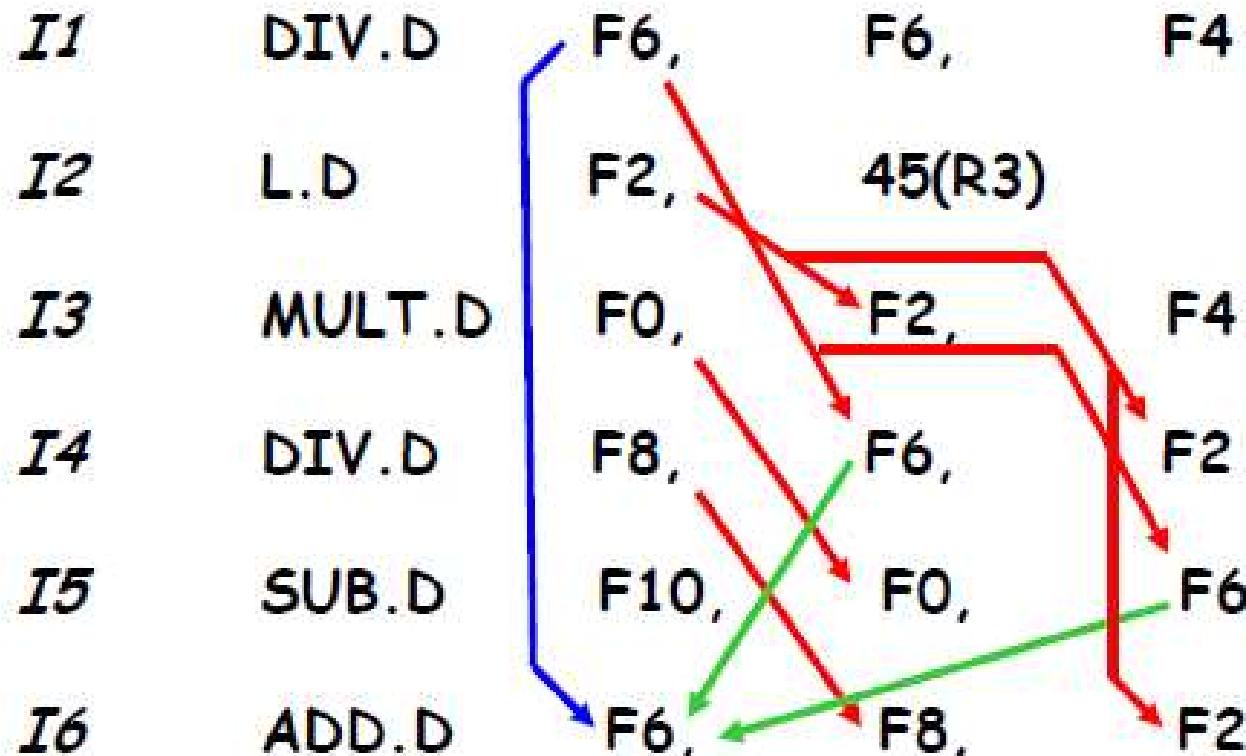
DIV.D F0,F2,F4  
ADD.D F10,F0,F8  
SUB.D F12,F8,F14

← Esta instr. não tem  
que esperar por DIV  
e por ADD

## Escalonamento dinâmico para diminuir o CPI

Um "stall" não bloqueia a execução de instruções em estágios posteriores do pipeline, nem bloqueia a emissão de novas instruções

# Conflitos de Dados - Exemplo

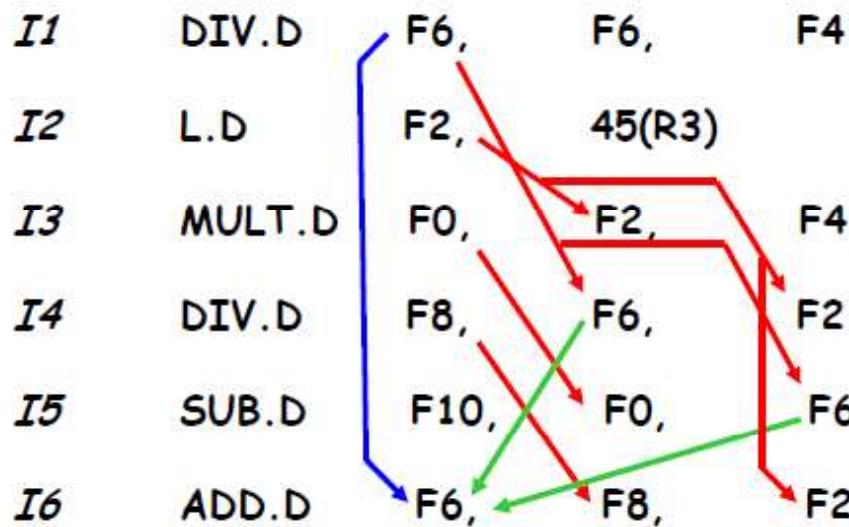


*Conflitos RAW*

*Conflitos WAR*

*Conflitos WAW*

# Escalonamento das instruções

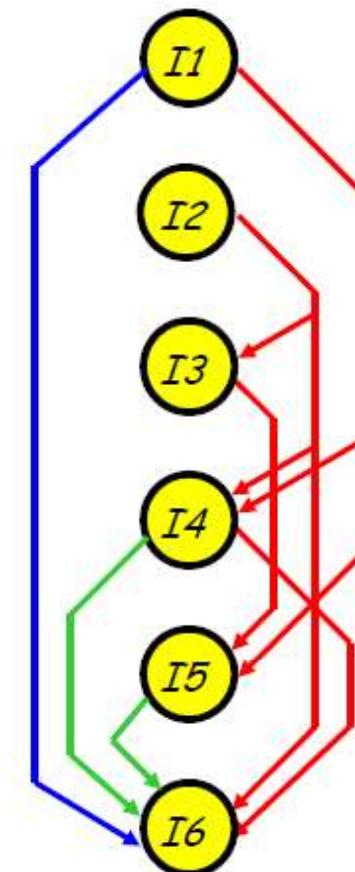


Ordenações válidas:

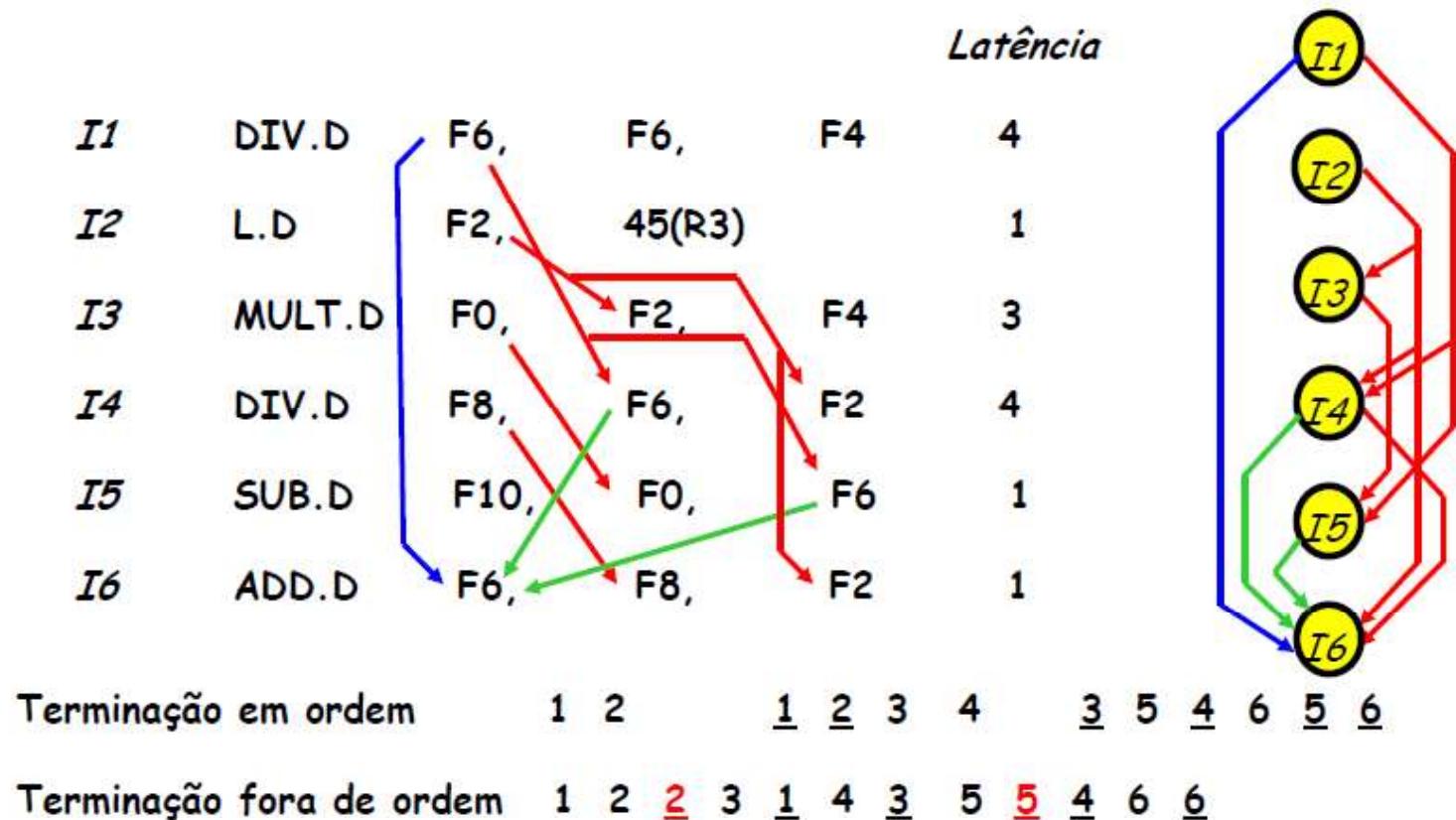
*em ordem*       $I_1$      $I_2$      $I_3$      $I_4$      $I_5$      $I_6$

*fora de ordem*     $\cancel{I_2}$      $\cancel{I_1}$      $I_3$      $I_4$      $I_5$      $I_6$

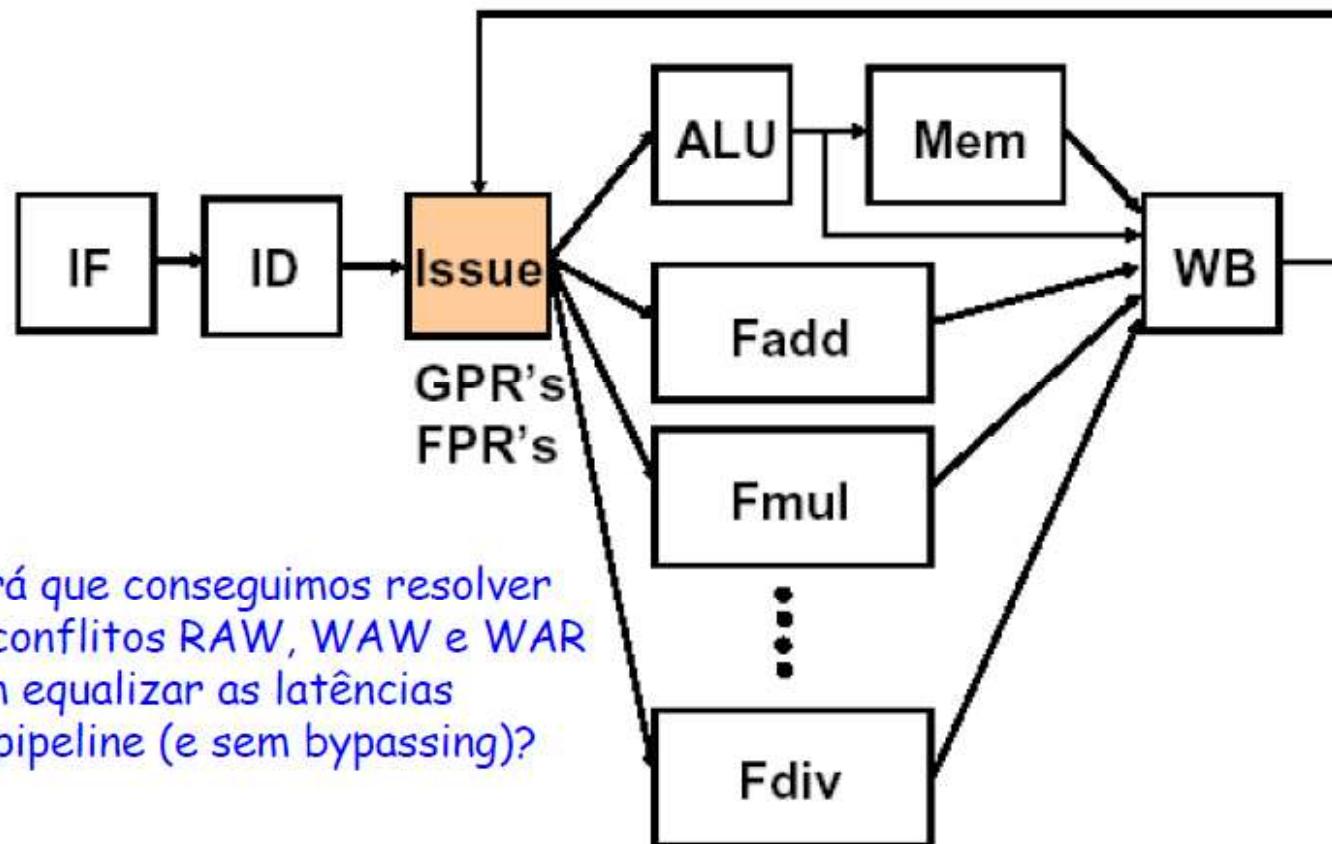
*fora de ordem*     $I_1$      $I_2$      $I_3$      $\cancel{I_5}$      $\cancel{I_4}$      $I_6$



Termino fora de ordem e emissão em ordem



# Latências diferentes



# Problemas

- Como impedir os conflitos WAR e WAW?
- Como acomodar diferentes latências?
  - Forwarding para os conflitos RAW muito mais complexos

Instrução	Ciclo de relógio																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
L.D F6,34(R2)	IF	ID	EX	MEM	WB												
L.D F2,45(R3)		IF	ID	EX	MEM	WB											
MULT.D F0,F2,F4			IF	ID	stall	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	MEM	WB
SUB.D F8,F6,F2				IF	ID	A1	A2	MEM	WB								
DIV.D F10,F0,F6					IF	ID	stall	D1	D2								
ADD.D F6,F8,F2						IF	ID	A1	A2	MEM	WB						

RAW

WAR

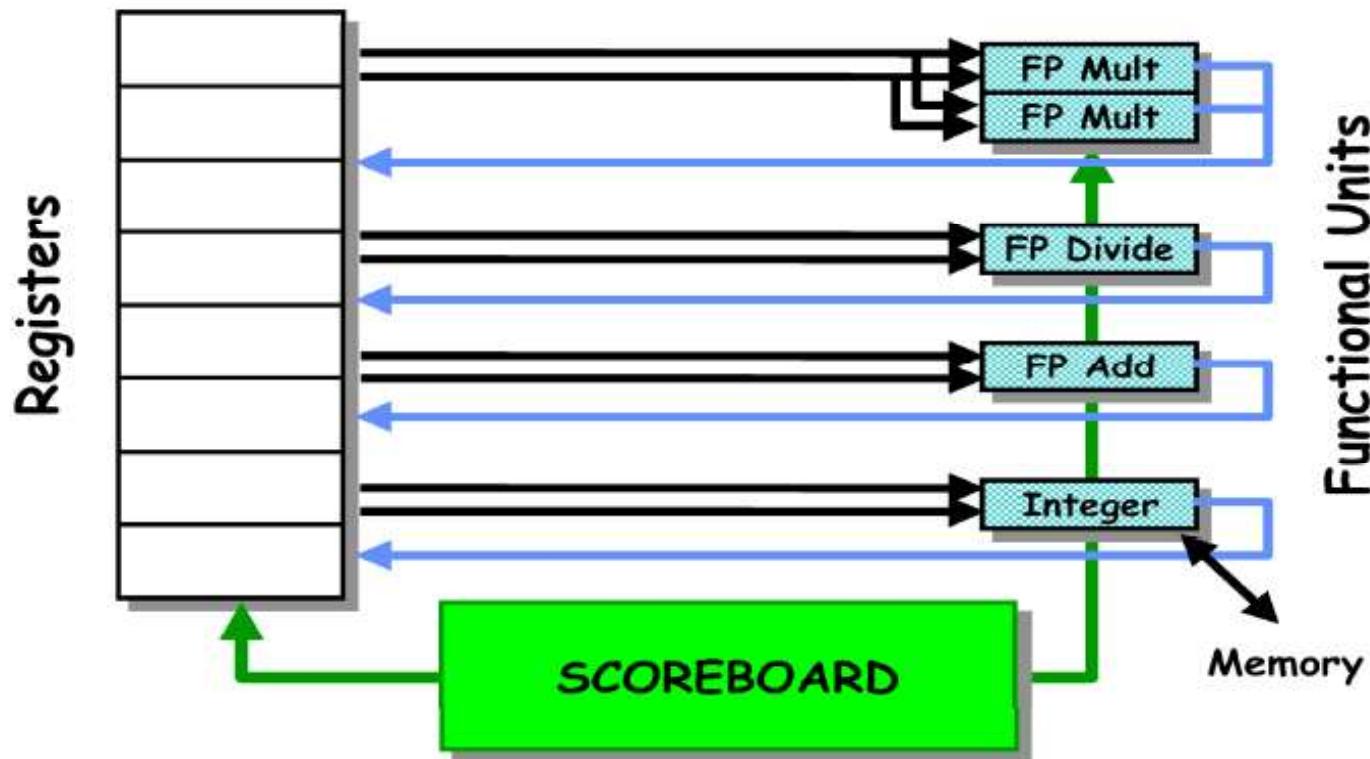
- Apareceu com o CDC 6600 em 1963
- As instruções são executadas desde que não dependam de instruções prévias e desde que não existam conflitos estruturais
- Não há “forwarding”
- Execução fora de ordem divide o estágio ID em:
  - Emite (“Issue”) — descodifica as instruções, identifica conflitos estruturais e WAW
  - Lê operandos (“Read Operands”) — espera até não haver conflitos de dados, em seguida lê operandos; resolve conflitos RAW



- A fast pipelined machine with 60-bit words (128 Kword main memory capacity, 32 banks)
- Ten functional units (parallel, unpipelined)
  - Floating Point adder, 2 multipliers,divider
  - Integer adder, 2 incrementers, ...
- Hardwired control (no microcoding)
- **Dynamic scheduling of instructions using a scoreboard**
- Ten Peripheral Processors for Input/Output

- A fast multi-threaded 12-bit integer ALU
- Very fast clock, 10 MHz (FP add in 4 clocks)
- > 400,000 transistors, 750 sq. ft., 5 tons, 150 kW, novel freon-based technology for cooling
- Fastest machine in world for 5 years (until 7600)
  - over 100 sold (\$7-10M each)

# Arquitetura de um Scoreboard simplificado



A máquina é pipelined, mas as unidades funcionais não são pipelined

# Implicações do Scoreboarding

- Termino fora de ordem → conflitos WAR, WAW?
- Solução para WAR:
  - Faz "stall" da escrita até os registradores terem sido lidos
- Solução para WAW:
  - Detecta conflito e faz "stall" da emissão de novas instruções até que as instruções anteriores estejam completas
- Para ser eficiente, precisa de ter múltiplas instruções em fase de execução → múltiplas unidades de execução, ou unidades de execução com pipeline, ou ambas
- O Scoreboard mantém o registro das dependências entre instruções que já foram emitidas

# Fases das instruções no Scoreboard

- Scoreboard substitui ID, EX e WB por 4 estágios
- Cada estágio corresponde a uma fase pela qual passa cada instrução

1 . **Emissão** — decodifica instrução e verifica se há conflitos estruturais ou WAW (ID1)

- Instruções emitidas pela ordem do programa → emissão em ordem
- Não emite instrução nesta fase
  - se existir um conflito estrutural
  - se existir um conflito WAW com uma instrução em execução (dependência de saída)
- Caso exista um destes conflitos, introduz "stalls" no pipeline

## 2 . Leitura dos operandos — espera até que não haja conflitos RAW e depois lê operandos (ID2)

- Os conflitos RAW são resolvidas nesta fase, ficando a instrução no scoreboard à espera de notificação da escrita do registrador (WB)
- Não há forwarding dos dados

## 3 . Execução dos operandos (EX)

- Unidade funcional inicia execução quando recebe os operandos
- Notifica o scoreboard quando termina a operação

# Fases das instruções no Scoreboard

4 . Escrita dos resultados — Termina a operação e vai escrever resultado no registrador de destino (WB)

- Notifica o scoreboard, que obriga a atrasar a escrita para resolver conflitos WAR com instruções à espera dos operandos

I1	DIV.D	F0,F1,F2	← Latência elevada do DIV
I2	SUB.D	F15,F0,F3	
I3	ADD.D	F3,F4,F5	

Em I2, F3 ainda não foi lido porque F0 não está disponível. Em I3, o Scoreboard retém a instrução no WB. F3 só é escrito quando I2 tiver lido F0 e F3

# Campos do scoreboard

- Estado da instrução:
  - Indica em qual das 4 fases se encontra a instrução
- Estado das unidades funcionais (FU):
  - 9 campos por FU
    - Busy: Indica se a unidade está ocupada
    - Op: Especifica a operação a realizar (e.g. + or -), na instrução
    - Fi: Número de registo destino, na instrução
    - Fj,Fk: Número dos registradores fonte, na instrução
    - Qj,Qk: Identifica as FUs que irão calcular os valores para os registradores Fj, Fk
    - Rj,Rk: Flags que indicam se Fj, Fk já têm o valor dos operandos
- Registro de estado do resultado
  - Indica qual a unidade funcional que escreve em cada um dos registradores → entrada vazia quando não há instruções para escrever naquele registo

# Controle do pipeline pelo scoreboard

Estado da instrução	Espera por	Operações
Emite instrução	Not busy (FU) and not result (D)	$\text{Busy(FU)} \leftarrow \text{Yes}; \text{Op(FU)} \leftarrow \text{op};$ $\text{Fi(FU)} \leftarrow \text{D}; \text{Fj(FU)} \leftarrow \text{S1};$ $\text{Fk(FU)} \leftarrow \text{S2}; \text{Qj} \leftarrow \text{Result(S1)};$ $\text{Qk} \leftarrow \text{Result(S2)}; \text{Rj} \leftarrow \text{not Qj};$ $\text{Rk} \leftarrow \text{not Qk}; \text{Result(D)} \leftarrow \text{FU}$
Lê operandos	$\text{Rj}$ and $\text{Rk}$	$\text{Rj} \leftarrow \text{No}; \text{Rk} \leftarrow \text{No}$
Execução completada	Unidade funcional terminar	
Escreve resultado	$\forall f((\text{Fj}(f) \neq \text{Fi(FU)} \text{ or } \text{Rj}(f) = \text{No}) \text{ &} (\text{Fk}(f) \neq \text{Fi(FU)} \text{ or } \text{Rk}(f) = \text{No}))$	$\forall f(\text{if } \text{Qj}(f) = \text{FU} \text{ then } \text{Rj}(f) \leftarrow \text{Yes});$ $\forall f(\text{if } \text{Qk}(f) = \text{FU} \text{ then } \text{Rk}(f) \leftarrow \text{Yes});$ $\text{Result}(\text{Fi(FU)}) \leftarrow 0; \text{Busy(FU)} \leftarrow \text{No}$

# Informações no Scoreboard - CR1

## Estado da instrução

Instrução	j	k	Lê Emite	EX Oper	Escreve final resultado
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		

## Estado das FU

Tempo restante	FU	Busy	Op	dest	S1	S2	FU	FU	Fj?	Fk?
	Int									
	Mult1									
	Mult2									
	Ad									
	Div									

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU									

# Atualização do Scoreboard - CR2

## Estado da instrução

Instrução	j	k	Lê	EX	Escreve	Emite	Oper final	resultado
L.D	F6	34+	R2		1			
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					

## Estado das FU

Tempo restante	FU	dest	S1	S2	FU	FU	Fj?	Fk?		
		Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Int		Y	LD	F6		R2			Y	
Mult1		N								
Mult2		N								
Ad		N								
Div		N								

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU				Int					

# Atualização do Scoreboard - CR3

## Estado da instrução

Instrução	j	k	Lê	EX	Escreve
			Emite	Oper final	resultado
L.D	F6	34+	R2	1	2
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		

Emite 2º L.D?

## Estado das FU

Tempo restante	FU	Busy	Op	dest	S1	S2	FU	FU	Fj?	Fk?
				Fi	Fj	Fk	Qj	Qk	Rj	Rk
Int		Y	LD	F6		R2				Y
Mult1		N								
Mult2		N								
Ad		N								
Div		N								

## Registro de estado dos resultados

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
						Int			

# Atualização do Scoreboard - CR4

## Estado da instrução

Instrução	j	k	Emite	Lê	EX	Escreve
				Oper	final	resultado
L.D	F6	34+	R2	1	2	3
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			

Emite 2º L.D?

## Estado das FU

Tempo restante	FU	Busy	Op	dest	S1	S2	FU	FU	Fj?	Fk?
				Fi	Fj	Fk	Qj	Qk	Rj	Rk
Int		Y	LD	F6		R2				N
Mult1		N								
Mult2		N								
Ad		N								
Div		N								

## Registro de estado dos resultados

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
					Int				

# Atualização do Scoreboard - CR5

## Estado da instrução

Instrução	j	k	Emite	Lê	EX	Escreve
			Oper	final	resultado	
L.D	F6	34+	R2	1	2	3
L.D	F2	45+	R3			4
MUL.D	F0	F2	F4			
SUB.D	F8	F6	F2			
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

Emite 2º L.D?

## Estado das FU

Tempo restante	FU	dest	S1	S2	FU	FU	Fj?	Fk?		
		Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Int		N								
Mult1		N								
Mult2		N								
Ad		N								
Div		N								

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU					Int				

# Atualização do Scoreboard - CR6

## Estado da instrução

Instrução	j	k	Emite	Lê	EX	Escreve
			Oper	final	resultado	
L.D	F6	34+	R2	1	2	3
L.D	F2	45+	R3	5		4
MUL.D	F0	F2	F4			
SUB.D	F8	F6	F2			
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

Emite 2º L.D

## Estado das FU

Tempo restante	FU	dest		S1	S2	FU	FU	Fj?	Fk?	
		Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Int		Y	LD	F2		R3			Y	
Mult1		N								
Mult2		N								
Ad		N								
Div		N								

## Registro de estado dos resultados

F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Int							

# Atualização do Scoreboard - CR7

## Estado da instrução

Instrução	j	k	Emite	Lê	EX	Escreve
			Oper	final	resultado	
L.D	F6	34+	R2	1	2	3
L.D	F2	45+	R3	5	6	
MUL.D	F0	F2	F4	6		
SUB.D	F8	F6	F2			
DIV.D	F10	F0	F6			
ADD.D	F6	F8	F2			

Emite MUL.D

## Estado das FU

Tempo restante	FU	Busy	Op	dest	S1	S2	FU	FU	Fj?	Fk?
				Fi	Fj	Fk	Qj	Qk	Rj	Rk
Int		Y	I.D	F2	R3				N	Y
Mult1		Y	MUL	F0	F2	F4	Int		N	Y
Mult2		N								
Ad		N								
Div		N								

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	Int							

# Atualização do Scoreboard - CR8a (Primeira metade do ciclo)

## Estado da instrução

Instrução	j	k	Lê	EX	Escreve
			Emite	Oper final	resultado
L.D	F6	34+	R2	1 2 3	4
L.D	F2	45+	R3	5 6 7	
MUL.D	F0	F2	F4	6	
SUB.D	F8	F6	F2	7	
DIV.D	F10	F0	F6		
ADD.D	F6	F8	F2		

Emite SUB.D

## Estado das FU

Tempo restante	FU	Busy	Op	dest	S1	S2	FU	FU	Fj?	Fk?
				Fi	Fj	Fk	Qj	Qk	Rj	Rk
Int		Y	LD	F2		R3			N	
Mult1		Y	MUL	F0	F2	F4	Int		N	Y
Mult2		N								
Ad		Y	SUB	F8	F6	F2		Int	Y	N
Div		N								

## Registo de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	Int			Ad				

# Atualização do Scoreboard - CR8a (Segunda metade do ciclo)

## Estado da instrução

Instrução	j	k	Lê	EX	Escreve
			Emite	Oper final	resultado
L.D	F6	34+	R2	1	2 3 4
L.D	F2	45+	R3	5	6 7
MUL.D	F0	F2	F4	6	
SUB.D	F8	F6	F2	7	
DIV.D	F10	F0	F6	8	
ADD.D	F6	F8	F2		

Emite DIV.D

## Estado das FU

Tempo restante	FU	Busy	Op	dest	S1	S2	FU	FU	Fj?	Fk?
				Fi	Fj	Fk	Qj	Qk	Rj	Rk
Int		Y	LD	F2		R3			N	
Mult1		Y	MUL	F0	F2	F4	Int		N	Y
Mult2		N								
Ad		Y	SUB	F8	F6	F2		Int	Y	N
Div		Y	DIV	F10	F0	F6	Mult1		N	Y

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	Int			Ad	Div			

# Atualização do Scoreboard - CR9)

## Estado da instrução

Instrução	j	k	Lê	EX	Escreve
			Emite	Oper final	resultado
L.D	F6	34+	R2	1 2 3	4
L.D	F2	45+	R3	5 6 7	8
MUL.D	F0	F2	F4	6	
SUB.D	F8	F6	F2	7	
DIV.D	F10	F0	F6	8	
ADD.D	F6	F8	F2		

## Estado das FU

Tempo restante	FU	Busy	Op	dest	S1	S2	FU	FU	Fj?	Fk?
				Fi	Fi	Fk	Qi	Qk	Ri	Rk
Int		N								
Mult1		Y	MUL	F0	F2	F4			Y	Y
Mult2		N								
Ad		Y	SUB	F8	F6	F2			Y	Y
Div		Y	DIV	F10	F0	F6	Mult1		N	Y

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1				Ad	Div			

# Atualização do Scoreboard - CR10

## Estado da instrução

Instrução	j	k	Emite	Lê Oper final	EX	Escreve resultado
L.D	F6	34+	R2	1	2	3 4
L.D	F2	45+	R3	5	6	7 8
MUL.D	F0	F2	F4	6	9	
SUB.D	F8	F6	F2	7	9	
DIV.D	F10	F0	F6	8		
ADD.D	F6	F8	F2			

Emite ADD.D?

## Estado das FU

Tempo restante	FU	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
10	Int	N								
	Mult1	Y	MUL	F0	F2	F4			Y	Y
	Mult2	N								
2	Ad	Y	SUB	F8	F6	F2			Y	Y
	Div	Y	DIV	F10	F0	F6	Mult1		N	Y

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1			Ad		Div			

# Atualização do Scoreboard - CR11

## Estado da instrução

Instrução	j	k	Emite	Lê Oper final	EX	Escreve resultado
L.D	F6	34+	R2	1	2	3 4
L.D	F2	45+	R3	5	6	7 8
MUL.D	F0	F2	F4	6	9	
SUB.D	F8	F6	F2	7	9	
DIV.D	F10	F0	F6	8		
ADD.D	F6	F8	F2			

Emite ADD.D?

## Estado das FU

Tempo restante	FU	Busy	Op	dest	S1	S2	FU	FU	Fj?	Fk?
				Fi	Fj	Fk	Qj	Qk	Rj	Rk
9	Int	N								
	Mult1	Y	MUL	F0	F2	F4			Y	Y
	Mult2	N								
1	Ad	Y	SUB	F8	F6	F2			Y	Y
	Div	Y	DIV	F10	F0	F6	Mult1		N	Y

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1				Ad	Div			

# Atualização do Scoreboard - CR12

## Estado da instrução

Instrução	j	k	Emite	Lê	EX	Escreve
			Oper	final	resultado	
L.D	F6	34+	R2	1	2	3 4
L.D	F2	45+	R3	5	6	7 8
MUL.D	F0	F2	F4	6	9	
SUB.D	F8	F6	F2	7	9	11
DIV.D	F10	F0	F6	8		
ADD.D	F6	F8	F2			

Emite ADD.D?

## Estado das FU

Tempo restante	FU	Busy	Op	dest	S1	S2	FU	FU	Fj?	Fk?
				Fi	Fj	Fk	Qj	Qk	Rj	Rk
8	Int	N							Y	Y
	Mult1	Y	MUL	F0	F2	F4				
	Mult2	N								
0	Ad	Y	SUB	F8	F6	F2			Y	Y
	Div	Y	DIV	F10	F0	F6	Mult1		N	Y

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1				Ad	Div			

# Atualização do Scoreboard - CR13

## Estado da instrução

Instrução	j	k	Emite	Lê Oper	EX final	Escreve resultado
L.D	F6	34+	R2	1	2	3
L.D	F2	45+	R3	5	6	7
MUL.D	F0	F2	F4	6	9	
SUB.D	F8	F6	F2	7	9	11
DIV.D	F10	F0	F6	8		12
ADD.D	F6	F8	F2			

Lê operandos de DIV.D?  
Emite ADD.D?

## Estado das FU

Tempo restante	FU	Busy	Op	dest	S1	S2	FU	FU	Fj?	Fk?
				Fi	Fj	Fk	Qj	Qk	Rj	Rk
7	Int	N								
	Mult1	Y	MUL	F0	F2	F4			Y	Y
	Mult2	N								
	Ad	N								
	Div	Y	DIV	F10	F0	F6	Mult1		N	Y

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1					Div			

# Atualização do Scoreboard - CR14

## Estado da instrução

Instrução	j	k	Emite	Lê Oper final	EX	Escreve resultado
L.D	F6	34+	R2	1	2	3 4
L.D	F2	45+	R3	5	6	7 8
MUL.D	F0	F2	F4	6	9	
SUB.D	F8	F6	F2	7	9	11 12
DIV.D	F10	F0	F6	o		
ADD.D	F6	F8	F2	13		

Emite ADD.D

## Estado das FU

Tempo restante	FU	Busy	Op	dest	S1 Fi	S2 Fj	FU Qj	FU Qk	Fj? Rj	Fk? Rk
6	Int	N							Y	Y
	Mult1	Y	MUL	F0	F2	F4				
	Mult2	N								
	Ad	Y	ADD	F6	F8	F2			Y	Y
	Div	Y	DIV	F10	F8	F6	Mult1		N	Y

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1			Ad		Div			

# Atualização do Scoreboard - CR15

## Estado da instrução

Instrução	j	k	Emite	Lê Oper final	EX	Escreve resultado
L.D	F6	34+	R2	1	2	3
L.D	F2	45+	R3	5	6	7
MUL.D	F0	F2	F4	6	9	
SUB.D	F8	F6	F2	7	9	11
DIV.D	F10	F0	F6	8		12
ADD.D	F6	F8	F2	13	14	

## Estado das FU

Tempo restante	FU	Busy	Op	dest	S1	S2	FU	FU	Fj?	Fk?
				Fi	Fj	Fk	Qj	Qk	Rj	Rk
5	Int	N								
	Mult1	Y	MUL	F0	F2	F4			Y	Y
	Mult2	N								
2	Ad	Y	ADD	F6	F8	F2			Y	Y
	Div	Y	DIV	F10	F0	F6	Mult1		N	Y

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1			Ad		Div			

# Atualização do Scoreboard - CR16

## Estado da instrução

Instrução	j	k	Emite	Lê Oper final	EX	Escreve resultado
L.D	F6	34+	R2	1	2	3
L.D	F2	45+	R3	5	6	7
MUL.D	F0	F2	F4	6	9	
SUB.D	F8	F6	F2	7	9	11
DIV.D	F10	F0	F6	8		
ADD.D	F6	F8	F2	13	14	

## Estado das FU

Tempo restante	FU	dest		S1	S2	FU	FU	Fj?	Fk?
		Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj
4	Int	N							
	Mult1	Y	MUL	F0	F2	F4			Y
	Mult2	N							Y
1	Ad	Y	ADD	F6	F8	F2			N
	Div	Y	DIV	F10	F0	F6	Mult1		Y

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1			Ad		Div			

# Atualização do Scoreboard - CR17

## Estado da instrução

Instrução	j	k	Emite	Lê Oper final	EX	Escreve resultado
L.D	F6	34+	R2	1	2	3
L.D	F2	45+	R3	5	6	7
MUL.D	F0	F2	F4	6	9	
SUB.D	F8	F6	F2	7	9	11
DIV.D	F10	F0	F6	8		
ADD.D	F6	F8	F2	13	14	16

**Conflito WAR!**  
**ADD.D não pode escrever em F6**

## Estado das FU

Tempo restante	FU	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
2	Int	N								
	Mult1	Y	MUL	F0	F2	F4			Y	Y
	Mult2	N								
	Ad	Y	ADD	F6	F8	F2			N	N
	Div	Y	DIV	F10	F0	F6	Mult1			Y

## Registro de estado dos resultados

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
Mult1				Ad		Div			

# Atualização do Scoreboard - CR19

## Estado da instrução

Instrução	j	k	Emite	Lê Oper final	EX	Escreve resultado
L.D	F6	34+	R2	1	2	3
L.D	F2	45+	R3	5	6	r
MUL.D	F0	F2	F4	6	9	19
SUB.D	F8	F6	F2	7	9	11
DIV.D	F10	F0	F6	8		
ADD.D	F6	F8	F2	13	14	16

## Estado das FU

Tempo restante	FU	dest		S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
		Busy	Op						
0	Int	N							
	Mult1	Y	MUL	F0	F2	F4		Y	Y
	Mult2	N							
	Ad	Y	ADD	F6	F8	F2		N	N
	Div	Y	DIV	F10	F0	F6	Mult1	N	Y

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1			Ad		Div			

# Atualização do Scoreboard - CR20

## Estado da instrução

Instrução	j	k	Emite	Lê	EX	Escreve final resultado
L.D	F6	34+	R2	1	2	3 4
L.D	F2	45+	R3	5	6	7
MUL.D	F0	F2	F4	6	9	19 20
SUB.D	F8	F6	F2	7	9	11 12
DIV.D	F10	F0	F6	8		
ADD.D	F6	F8	F2	13	14	16

## Estado das FU

Tempo restante	FU	dest		S1	S2	FU	FU	Fj?	Fk?	
		Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Int		N								
Mult1		N								
Mult2		N								
Ad		Y	ADD	F6	F8	F2			N	N
Div		Y	DIV	F10	F0	F6			Y	Y

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU					Ad		Div		

# Atualização do Scoreboard - CR21

## Estado da instrução

Instrução	j	k	Emite	Lê Oper final	EX	Escreve resultado
L.D	F6	34+	R2	1	2	3
L.D	F2	45+	R3	5	6	7
MUL.D	F0	F2	F4	6	9	19
SUB.D	F8	F6	F2	7	9	11
DIV.D	F10	F0	F6	8	21	12
ADD.D	F6	F8	F2	13	11	16

WAR resolvido

## Estado das FU

Tempo restante	FU	dest		S1 Fj	S2 Fk	FU Qj	FU Qk	Fj? Rj	Fk? Rk
		Busy	Op						
Int		N							
Mult1		N							
Mult2		N							
Ad		Y	ADD	F6	F8	F2		N	N
Div		Y	DIV	F10	F0	F6		Y	Y

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU				Ad		Div			

# Atualização do Scoreboard - CR22

## Estado da instrução

Instrução	j	k	Emite	Lê Oper final	EX	Escreve resultado
L.D	F6	34+	R2	1	2	3 4
L.D	F2	45+	R3	5	6	7 8
MUL.D	F0	F2	F4	6	9	19 20
SUB.D	F8	F6	F2	7	9	11 12
DIV.D	F10	F0	F6	8	21	
ADD.D	F6	F8	F2	13	14	16 22

## Estado das FU

Tempo restante	FU	Busy	Op	dest	S1	S2	FU	FU	Fj?	Fk?
	Int	N								
	Mult1	N								
	Mult2	N								
	Ad	N								
39	Div	Y	DIV	F10	F0	F6			N	N

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU							Div		

# Atualização do Scoreboard - CR61

## Estado da instrução

Instrução	j	k	Emite	Lê Oper final	EX	Escreve resultado
L.D	F6	34+	R2	1	2	3 4
L.D	F2	45+	R3	5	6	7 8
MUL.D	F0	F2	F4	6	9	19 20
SUB.D	F8	F6	F2	7	9	11 12
DIV.D	F10	F0	F6	8	21	61
ADD.D	F6	F8	F2	13	14	10 22

## Estado das FU

Tempo restante	FU	Busy	Op	dest	S1 Fi	S2 Fj	FU Qj	FU Qk	Fj? Rj	Fk? Rk
	Int	N								
	Mult1	N								
	Mult2	N								
	Ad	N								
0	Div	Y	DIV	F10	F0	F6			N	N

## Registro de estado dos resultados

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
									Div

# Atualização do Scoreboard - CR62

## Estado da instrução

Instrução	j	k	Emite	Lê Oper	EX final	Escreve resultado
L.D	F6	34+	R2	1	2	3
L.D	F2	45+	R3	5	6	7
MUL.D	F0	F2	F4	6	9	19
SUB.D	F8	F6	F2	7	9	11
DIV.D	F10	F0	F6	8	21	61
ADD.D	F6	F8	F2	13	14	16

## Estado das FU

Tempo restante	FU	Busy	dest	S1	S2	FU	FU	Fj?	Fk?
			Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Int	N							
	Mult1	N							
	Mult2	N							
	Ad	N							
	Div	N							

## Registro de estado dos resultados

FU	F0	F2	F4	F6	F8	F10	F12	...	F30

# Resultado final no Scoreboard

## Estado da instrução

Instrução	j	k	Emite	Lê Oper final	EX	Escreve resultado	
L.D	F6	34+	R2	1	2	3	+
L.D	F2	45+	R3	5	7	8	
MUL.D	F0	F2	F4	6	9	19	
SUB.D	F8	F6	F2	7	9	11	12
DIV.D	F10	F0	F6	8	21	61	62
ADD.D	F6	F8	F2	13	14	16	22

Emissão em ordem

Execução e termino  
fora de ordem

## Estado das FU

Tempo restante	FU	dest	S1	S2	FU	FU	Fj?	Fk?
		Busy	Op	Fi	Fj	Fk	Qj	Qk
Int		N						
Mult1		N						
Mult2		N						
Ad		N						
Div		N						

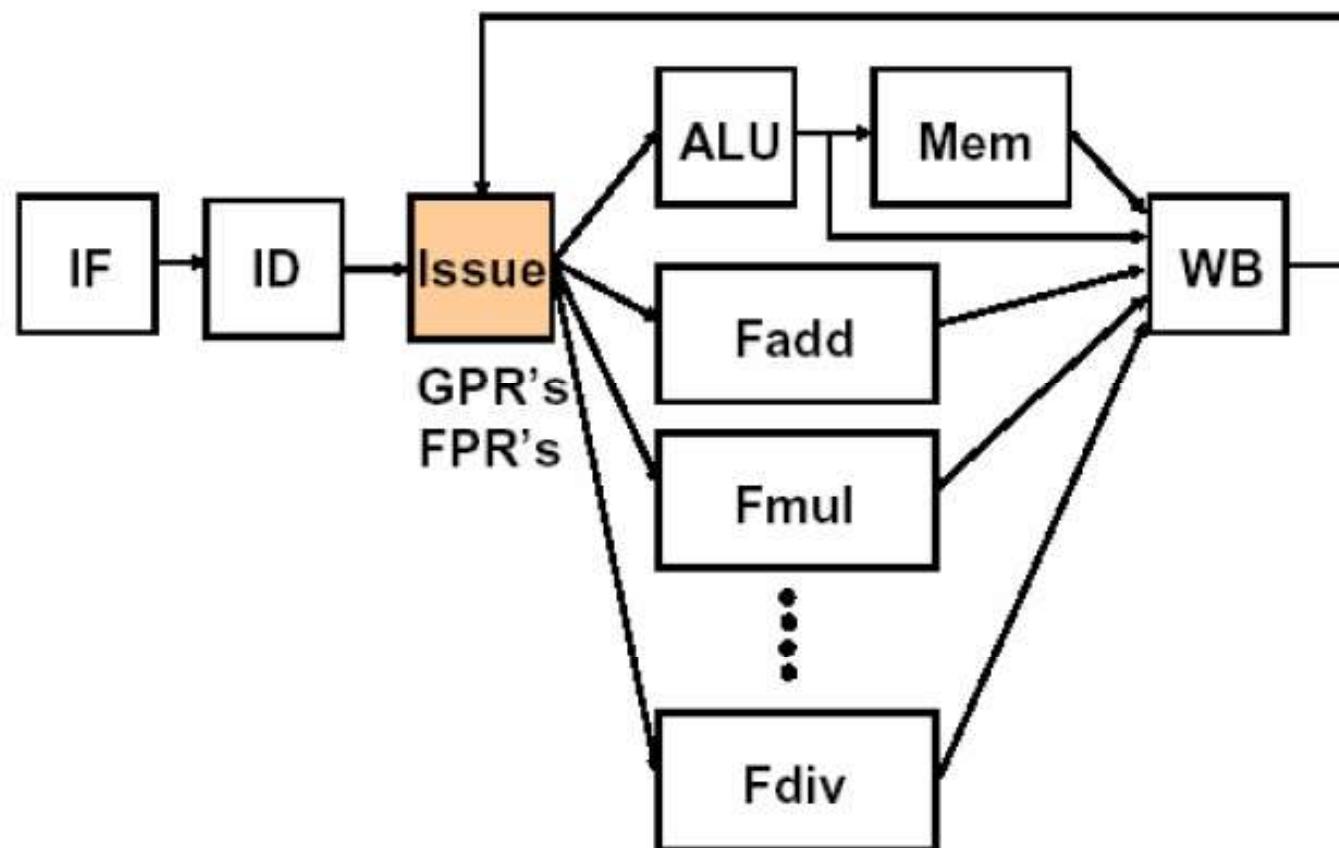
## Registro de estado dos resultados

FU	F0	F2	F4	F6	F8	F10	F12	...	F30

# Limitações do Scoreboarding

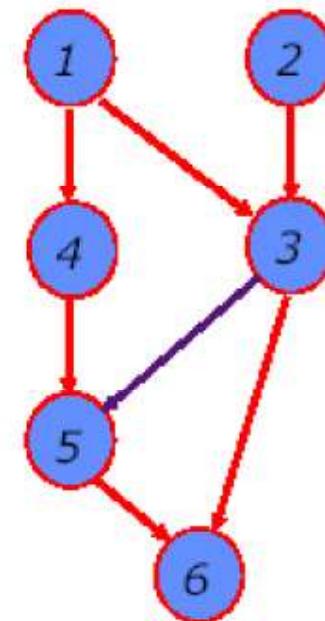
- Forwarding difícil de aplicar
  - Mas resultados disponíveis assim que os registos são escritos no WB
- Controle centralizado
- Estrutura complexa
  - Pode implicar múltiplas atualizações num único ciclo de relógio (exemplo em CR8)
- Conflitos WAW bloqueiam o pipeline
- Conflitos WAR atrasam escritas nos registos
- Não há emissão de instruções se existirem conflitos estruturais
- Outras técnicas de agendamento dinâmico das instruções?

## In-Order Issue Pipeline



# Limitações Emissão de instruções por ordem

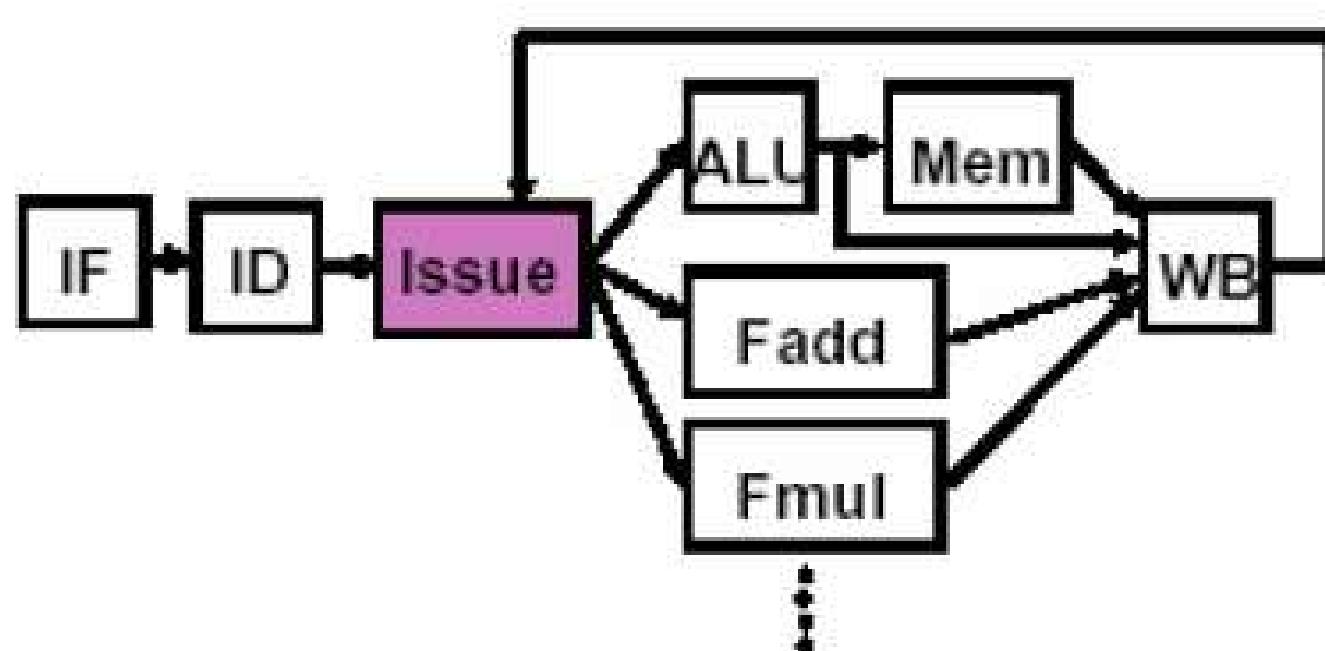
1	LD	F2,	34(R2)	latency 1
2	LD	F4,	45(R3)	long
3	MULTD	F6,	F4, F2	3
4	SUBD	F8,	F2, F2	1
5	DIVD	F4,	F2, F8	4
6	ADDD	F10,	F6, F4	1



1 (2,1) . . . . 2 3 4 4 3 5 . . . 5 6 6

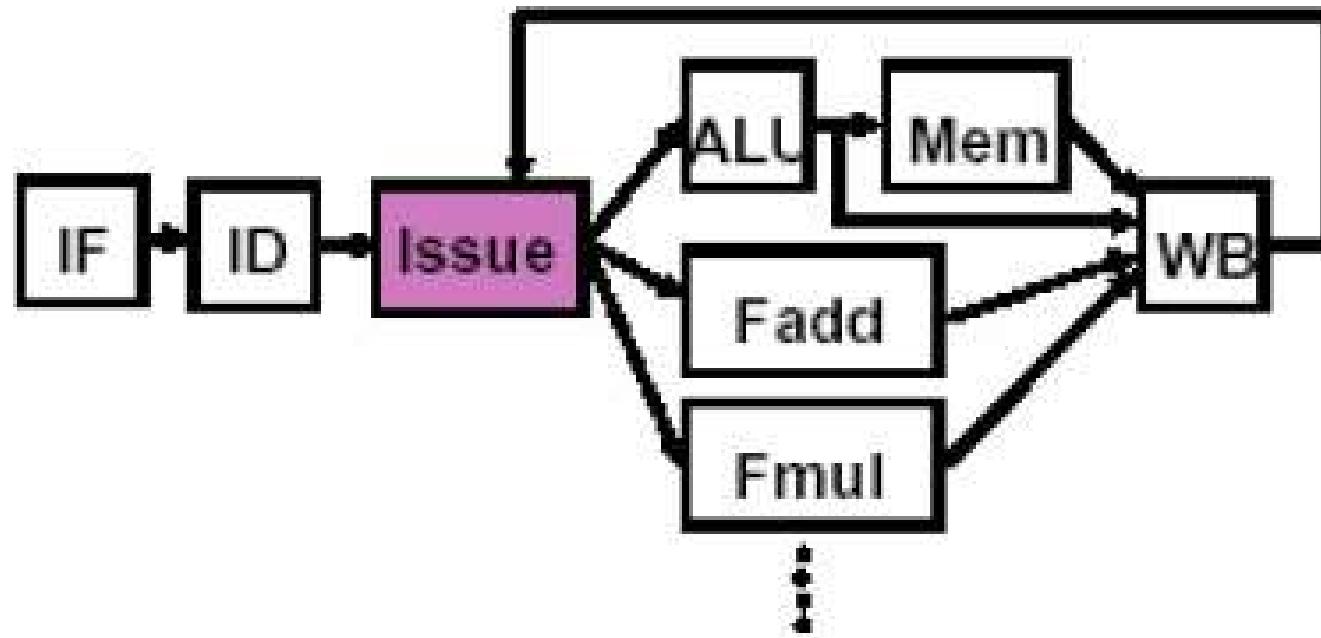
Pelo fato da emissão de instruções ser realizada por ordem  
a instrução 4 não pode ser emitida aqui

# Emissão fora de ordem



- O estágio de emissão ("issue") atua como um buffer que guarda várias instruções, prontas a serem emitidas
- O estágio ID junta mais de uma instrução ao andar da emissão se houver espaço e se a instrução não provocar um conflito WAR ou WAW

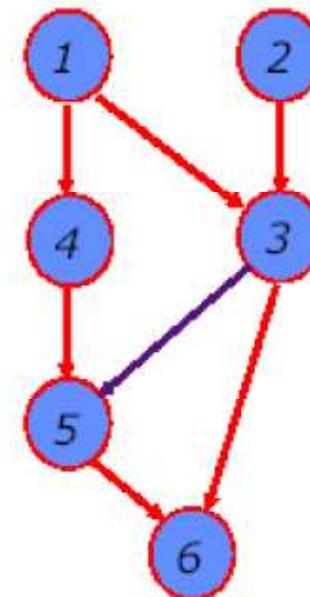
# Emissão fora de ordem



- Pode ser emitida qualquer instrução no buffer cujos conflitos RAW tenham sido resolvidos (considerando a emissão de uma instrução por ciclo de relógio).
- No WB, novas instruções são avaliadas

# Emissão fora de ordem

1	LD	F2,	34(R2)	<i>latency</i> 1
2	LD	F4,	45(R3)	<i>long</i>
3	MULTD	F6,	F4, F2	3
4	SUBD	F8,	F2, F2	1
5	DIVD	F4,	F2, F8	4
6	ADDD	F10,	F6, F4	1



Por ordem: 1 (2,1) . . . . . 2 3 4 4 3 5 . . . 5 6 6

Fora de ordem: 1 (2,1) 4 4 . . . . 2 3 . . 3 5 . . . 5 6 6

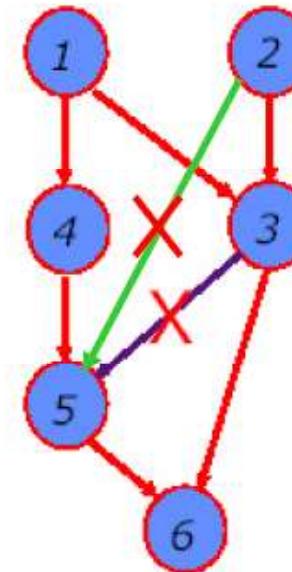
A emissão fora de ordem, por si só, não garante ganhos de desempenho significativos

# Limitação devido ao número de registradores

- O número de registradores do ISA limita o número de instruções que podem coexistir, em cada instante, num pipeline complexo
  - Não conseguimos manter cheio um pipeline de FP (ponto flutuante) se o número de FPRs (registradores de ponto flutuante) for reduzido
    - Ex: os FPRs da família IBM 360 eram em número de 4
- Podemos aumentar o número de registradores disponíveis para o ISA sem perda de compatibilidade?
  - Robert Tomasulo propôs uma solução engenhosa baseada no conceito de renomeação dinâmica dos registradores ("register renaming")
- ILP (Instruction Level Parallelism) dinâmico

# ILP com renomeação

1	LD	F2,	34(R2)	latency 1
2	LD	F4,	45(R3)	long
3	MULTD	F6,		
4	SUBD	F8,	F2, F2	3
5	DIVD		F2, F8	1
6	ADDD	F10,	F6, F4'	4



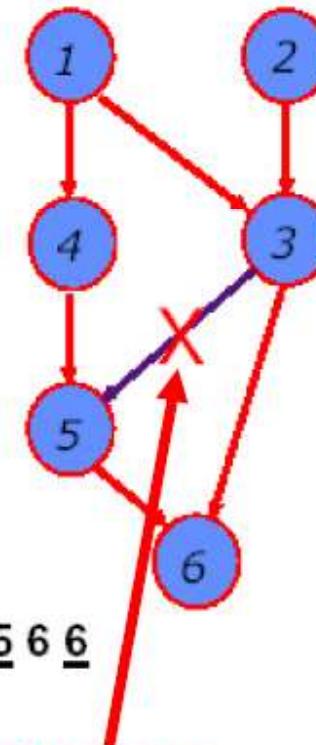
Por ordem: 1 (2,1) . . . . . 2 3 4 4 (5,3) . . . 5 6 6

Fora de ordem: 1 (2,1) 4 4 5 . . . 2 (3,5) 3 6 6

Dependências WAW e WAR são eliminadas com a Renomeação

# ILP com renomeação

1	LD	F2,	34(R2)	latency 1
2	LD	F4,	45(R3)	long
3	MULTD	F6,	F4,	F2
4	SUBD	F8,	F2,	F2
5	DIVD	<b>F4'</b> ,	F2,	F8
6	ADDD	F10,	F6,	<b>F4'</b>



Por ordem: 1 (2,1) . . . . . 2 3 4 4 (5,3) . . . 5 6 6

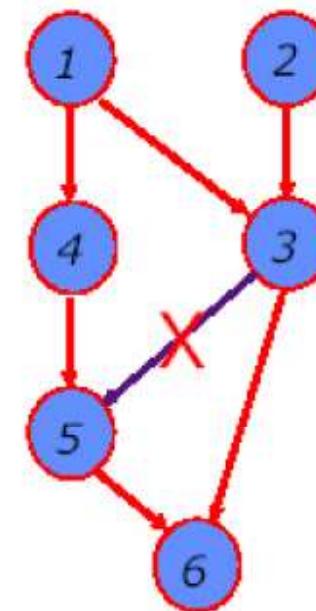
Fora de ordem: 1 (2,1) 4 4 5 . . . 2 (3,5) 3 6 6

Com **Renomeação**, a instrução 5 pode ser emitida imediatamente.

Sem **Renomeação**, a instrução 5 só podia ser emitida depois da instrução 2 ter terminado e da instrução 3 ter adquirido os operandos

# ILP com renomeação

1	LD	F2,	34(R2)	latency 1
2	LD	F4,	45(R3)	long
3	MULTD	F6,	F4,	3
4	SUBD	F8,	F2,	F2 1
5	DIVD	F4',	F2,	F8 4
6	ADDD	F10,	F6,	F4' 1

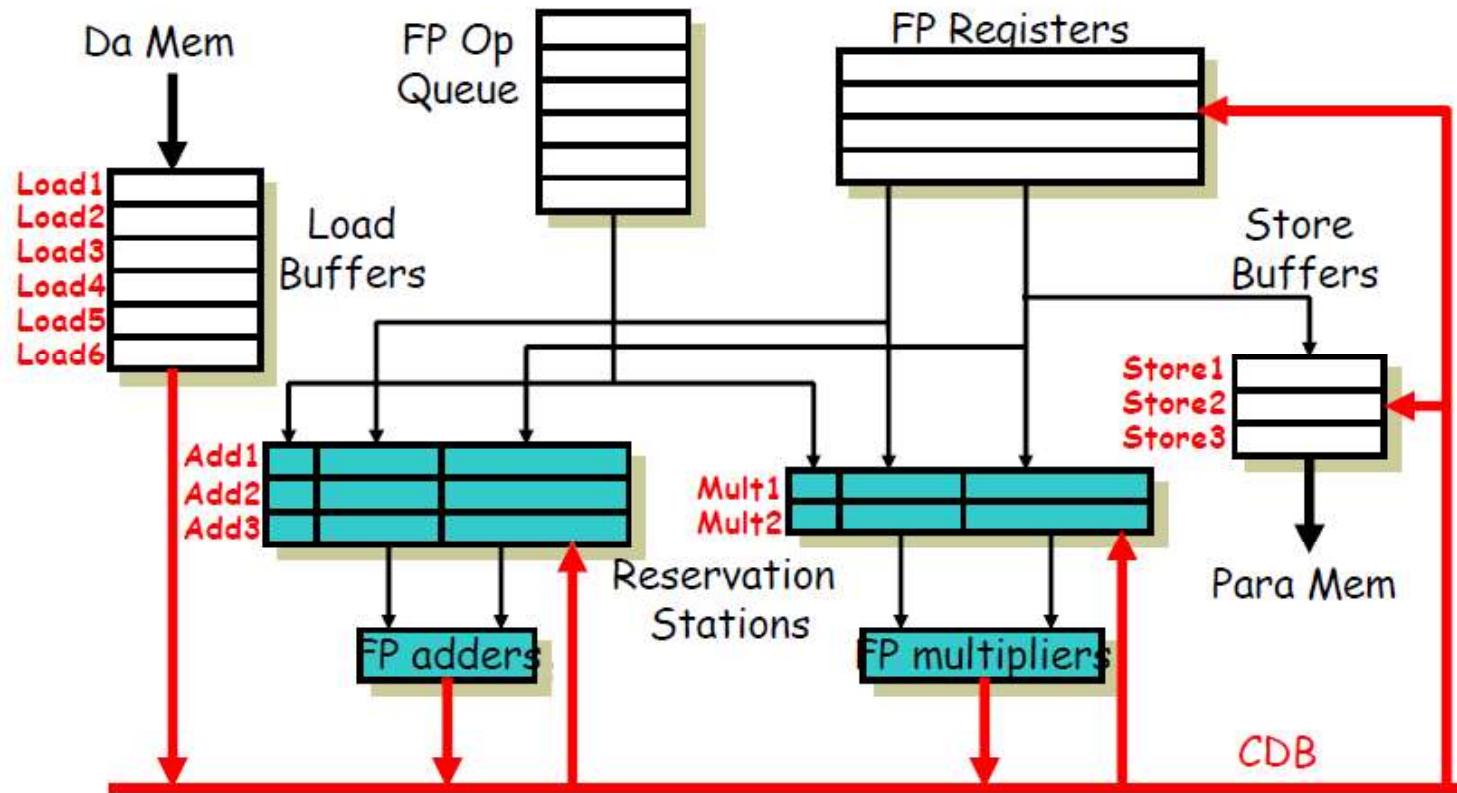


Renomeação implica mais Registradores.  
Será que pode ser implementada em hardware?

# Escalonamento dinâmico - algoritmo de Tomasulo

- Para o IBM 360/91, cerca de 3 anos depois do CDC 6600 (1966)
- Porquê?
  - Latências elevadas nas instruções de acesso à memória e nas operações de ponto flutuante
  - A maioria das instruções usavam operandos em memória (a máquina era CISC)
  - Apenas 4 FPRs contra 8 no CDC 6600)
- Importância do algoritmo de Tomasulo?
  - Usado no Alpha 21264, HP 8000, MIPS 10000, Pentium II, PowerPC 604, ...

# Organização de Tomasulo



# Tomasulo vs scoreboard

- Controle e buffers distribuídos pelas Unidades Funcionais (FU) vs. controle e buffers centralizados no scoreboard
  - Buffers das Unidades Funcionais (FUs) designados por “reservation stations” (RS) ou estações de reserva, que guardam os operandos das instruções a executar em cada FU
- Número dos Registradores nas instruções substituídos por valores ou ponteiros para as estações de reserva; ”register renaming”
  - Impede conflitos WAR e WAW
  - Mais estações de reserva do que registradores → podem-se fazer optimizações que os compiladores não conseguem
- Resultados vão das RS para as FU (mas não através dos registradores), e destas últimas para o Common Data Bus (CDB), que faz o broadcast dos resultados para todas as RS
- Load e Stores tratados como FU com RS

# Esquema de Tomasulo

- Uma instrução é emitida desde que haja estações de reserva livres para a FU pretendida:
  - Os valores disponíveis dos registradores fonte são copiados para a estação de reserva → RENOMEAÇÃO
  - Quando o valor de um registrador fonte não está disponível, é passada para a estação de reserva indicação da RS que vai gerar esse valor
- Quando uma FU calcula um resultado, escreve-o (faz o broadcast) no Barramento de Dados Comum (CDB) para as Estações de reserva que dele necessitam o poderem ir buscar (sem passagem pelo registrador de destino)

# Características do esq. de Tomasulo

- Os conflitos WAW são facilmente resolvidos
  - Indicando para cada registrador a estação de reserva que produz o resultado mais recente
- Unidades de leitura/escrita na memória (loads e stores) podem ser consideradas como FUs com estações de reserva
- O esquema de Tomasulo implementa uma espécie de forwarding
  - um resultado colocado no CDB é imediatamente visto por todas as estações de reserva
- O número de estações de reserva pode ser maior que o número de registradores
  - Fornece-se um conjunto grande de registradores virtuais para renomeação

# Fases do esquema de Tomasulo

- **Emite** — a instrução que está na no topo da FP Op Queue, se houver uma estação de reserva livre para a FU requerida
  - Se não, há um conflito estrutural e o pipeline tem que parar
  - Se houver, a instrução é emitida para a estação de reserva
    - com o valor dos operandos disponíveis nos registradores fonte, se existirem
    - com indicação das FU/estações de reserva que se encontram a calcular os operandos não disponíveis nos registros fonte
  - A instrução na RS renomeia os registradores, eliminando conflitos WAR e WAW

# Fases do esquema de Tomasulo

- **Executa** — executa as operações (EX)
  - Caso haja pelo menos um operando não acessível → RS verifica continuamente no CDB se já foi calculado
  - Quando um resultado (operando) é colocado no CDB o seu valor é copiado para as estações de reserva correspondentes
  - Quando todos os operandos estão acessíveis numa estação de reserva, inicia-se a execução da instrução
  - Como a execução da instrução só é feita depois dos dois operandos estarem disponíveis, eliminam-se conflitos RAW

# Fases do esquema de Tomasulo

- **Escreve resultado** — A FU terminou a operação e vai colocar o resultado no CDB
  - Além do resultado é colocada também a etiqueta da estação de reserva que lhe deu origem
  - Quando a etiqueta da estação de reserva ou associada a um registrador coincide com a etiqueta presente no CDB, o valor é imediatamente colocado no registrador e na estação de reserva
- Os buffers de escrita na memória podem ser considerados também como estações de reserva
  - Escreve-se o valor no buffer de escrita que, por sua vez, quando o endereço e os dados estão acessíveis, sinaliza a unidade de memória para efetuar a escrita

# Campos associados às RSs

- Cada estação de reserva tem 7 campos
  - Busy: Indica se estação de reserva (e FU) está ocupada
  - Op: Especifica a operação a realizar (ex., + or -)
  - Vj,Vk: Valor dos operandos
  - Qj,Qk: Etiquetas que indicam as RSs que irão calcular os valores dos operandos; um valor 0 na etiqueta indica que um operando está disponível (em Vj,Vk) ou que não é necessário
  - A: Para Load/Store, guarda o valor do endereço efetivo em memória
- Cada registrador tem um campo adicional
  - Qi: Etiqueta da estação de reserva que irá calcular o valor a ser colocado neste registrador

# Informação nas RSs

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)
L.D	F6	34+	R2			LD1	N
L.D	F2	45+	R3			LD2	N
MUL.D	F0	F2	F4			LD3	N
SUB.D	F8	F6	F2				
DIV.D	F10	F0	F6				
ADD.D	F6	F8	F2				

## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
Add1				N			
Add2				N			
Add3				N			
Mult1				N			
Mult2				N			

## Registro de estado dos resultados

F0	F2	F4	F6	F8	F10	F12	...	F30
FU								

# Algoritmo de Tomasulo – CR1

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)
L.D	F6	34+	R2	1		LD1	Y 34+R2
L.D	F2	45+	R3			LD2	N
MUL.D	F0	F2	F4			LD3	N
SUB.D	F8	F6	F2				
DIV.D	F10	F0	F6				
ADD.D	F6	F8	F2				

## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 V <sub>k</sub>	RS Qj	RS Q <sub>k</sub>
Add1		N					
Add2		N					
Add3		N					
Mult1		N					
Mult2		N					

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU				LD1					

# Algoritmo de Tomasulo – CR2

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado
L.D	F6	34+	R2	1	
L.D	F2	45+	R3	2	
MUL.D	F0	F2	F4		
SUB.D	F8	F6	F2		
DIV.D	F10	F0	F6		
ADD.D	F6	F8	F2		

	Busy	Ender (A)
LD1	Y	J4+R2
LD2	Y	45+R3
LD3	N	

## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
Add1				N			
Add2				N			
Add3				N			
Mult1				N			
Mult2				N			

Ao contrário do CDC 6600, podem existir vários Loads em simultâneo

## Registro de estado dos resultados

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
		LD2			LD1				

# Algoritmo de Tomasulo – CR3

## Estado da instrução

Instrução	j	k	Emite	EX	Escreve resultado
L.D	F6	34+	R2	1	3
L.D	F2	45+	R3	2	
MUL.D	F0	F2	F4	3	
SUB.D	F8	F6	F2		
DIV.D	F10	F0	F6		
ADD.D	F6	F8	F2		

Busy	Ender (A)
LD1	Y      34+R2
LD2	Y      45+R3
LD3	N

Registo F2

Renomeado  
na estação de  
reserva.  
Operando F4

## Estações de reserva

Tempo restante	RS	Busy	Op	S1	S2	RS	RS
				Vj	Vk	Qj	Qk
Add1		N					
Add2		N					
Add3		N					
Mult1		Y	MUL.D		R(F4)	LD2	
Mult2		N					

no Registrador  
LD1 completado  
no fim deste CR.  
Quem precisa  
de F6?

## Registro de estado dos resultados

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Mult1	LD2		LD1					

# Algoritmo de Tomasulo – CR4

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado
L.D	F6	34+	R2	1 2	3 4
L.D	F2	45+	R3	4	
MUL.D	F0	F2	F4	3	
SUB.D	F8	F6	F2	4	
DIV.D	F10	F0	F6		
ADD.D	F6	F8	F2		

Busy	Ender (A)
LD1	N
LD2	Y
LD3	45+R3

## Estações de reserva

Tempo restante	RS	Busv	Op	S1	S2	RS	RS
				Vi	Vk	Q1	Qk
Add1	Y	SUB.D	M(A1)			LD2	
Add2	N						
Add3	N						
Mult1	Y	MUL.D		R(F4)		LD2	
Mult2	N						

Operando F6 em memória.  
LD2 completado.  
Quem precisa de F2?

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	LD2		M(A1)	Add1				

# Algoritmo de Tomasulo – CR5

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)	
L.D	F6	34+	R2	1	3	4	LD1	N
L.D	F2	45+	R3	2	4	5	LD2	N
MUL.D	F0	F2	F4	3			LD3	N
SUB.D	F8	F6	F2	4				
DIV.D	F10	F0	F6	5				
ADD.D	F6	F8	F2					

## Estações de reserva

Tempo restante	RS	Busv	Op	S1 Vi	S2 V <sub>k</sub>	RS Qi	RS Ok
2	Add1	Y	SUB.D	M(A1)	M(A2)		
	Add2	N					
10	Add3	N					
	Mult1	Y	MUL.D	M(A2)	R(F4)		
	Mult2	Y	DIV.D		M(A1)	Mult1	

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	M(A2)		M(A1)	Add1	Mult2			

# Algoritmo de Tomasulo – CR6

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)	
L.D	F6	34+	R2	1	3	4	LD1	N
L.D	F2	45+	R3	2	4	5	LD2	N
MUL.D	F0	F2	F4	3			LD3	N
SUB.D	F8	F6	F2	4				
DIV.D	F10	F0	F6	5				
ADD.D	F6	F8	F2	6				

## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
1	Add1	Y	SUB.D	M(A1) M(A2)			
	Add2	Y	ADD.D		M(A2)	Add1	
	Add3	N					
9	Mult1	Y	MUL.D	M(A2)	R(F4)		
	Mult2	Y	DIV.D		M(A1)	Mult1	

## Registro de estado dos resultados

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Mult1	M(A2)		Add2	Add1	Mult2			

# Algoritmo de Tomasulo – CR7

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)	
L.D	F6	34+	R2	1	3	4	LD1	N
L.D	F2	45+	R3	2	4	5	LD2	N
MUL.D	F0	F2	F4	3			LD3	N
SUB.D	F8	F6	F2	4	7			
DIV.D	F10	F0	F6	5				
ADD.D	F6	F8	F2	6				

## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
0	Add1	Y	SUB.D	M(A1) M(A2)			
	Add2	Y	ADD.D		M(A2)	Add1	
	Add3	N					
8	Mult1	Y	MUL.D	M(A2)	R(F4)		
	Mult2	Y	DIV.D		M(A1)	Mult1	

Add1 acabou.  
Quem precisa  
de F8?

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	M(A2)		Add2	Add1	Mult2			

# Algoritmo de Tomasulo – CR8

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)	
L.D	F6	34+	R2	1	3	4	LD1	N
L.D	F2	45+	R3	2	4	5	LD2	N
MUL.D	F0	F2	F4	3			LD3	N
SUB.D	F8	F6	F2	4	7	8		
DIV.D	F10	F0	F6	5				
ADD.D	F6	F8	F2	6				

## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
2	Add1	N					
2	Add2	Y	ADD.D	(M-M)	M(A2)		
2	Add3	N					
7	Mult1	Y	MUL.D	M(A2)	R(F4)		
7	Mult2	Y	DIV.D		M(A1)	Mult1	

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	M(A2)		Add2	(M-M)	Mult2			

# Algoritmo de Tomasulo – CR9

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)	
L.D	F6	34+	R2	1	3	4	LD1	N
L.D	F2	45+	R3	2	4	5	LD2	N
MUL.D	F0	F2	F4	3			LD3	N
SUB.D	F8	F6	F2	4	7	8		
DIV.D	F10	F0	F6	5				
ADD.D	F6	F8	F2	6				

## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	N					
1	Add2	Y	ADD.D	(M-M)	M(A2)		
	Add3	N					
6	Mult1	Y	MUL.D	M(A2)	R(F4)		
	Mult2	Y	DIV.D		M(A1)	Mult1	

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	M(A2)		Add2	(M-M)	Mult2			

# Algoritmo de Tomasulo – CR10

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)	
L.D	F6	34+	R2	1	3	4	LD1	N
L.D	F2	45+	R3	2	4	5	LD2	N
MUL.D	F0	F2	F4	3			LD3	N
SUB.D	F8	F6	F2	4	7	8		
DIV.D	F10	F0	F6	5				
ADD.D	F6	F8	F2	6	10			

## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	N					
0	Add2	Y	ADD.D	(M-M) M(A2)			
	Add3	N					
5	Mult1	Y	MUL.D	M(A2) R(F4)			
	Mult2	Y	DIV.D		M(A1) Mult1		

Add2 acabou.  
Quem precisa  
de F6?

## Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	M(A2)		Add2	(M-M)	Mult2			

# Algoritmo de Tomasulo – CR11

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)	
L.D	F6	34+	R2	1	3	4	LD1	N
L.D	F2	45+	R3	2	4	5	LD2	N
MUL.D	F0	F2	F4	3			LD3	N
SUB.D	F8	F6	F2	4	7	8		
DIV.D	F10	F0	F6	5				
ADD.D	F6	F8	F2	6	10	11		

## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	N					
	Add2	N					
	Add3	N					
4	Mult1	Y	MUL.D	M(A2)	R(F4)		
	Mult2	Y	DIV.D		M(A1)	Mult1	

Só restam as instruções com latência elevada

## Registro de estado dos resultados

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

# Algoritmo de Tomasulo – CR12

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)
L.D	F6	34+	R2	1	3	4	LD1 N
L.D	F2	45+	R3	2	4	5	LD2 N
MUL.D	F0	F2	F4	3			LD3 N
SUB.D	F8	F6	F2	4	7	8	
DIV.D	F10	F0	F6	5			
ADD.D	F6	F8	F2	6	10	11	

## Estações de reserva

Tempo restante	RS	S1		S2		RS Qj	RS Qk
		Busy	Op	Vj	Vk		
	Add1	N					
	Add2	N					
	Add3	N					
3	Mult1	Y	MUL.D	M(A2)	R(F4)		
	Mult2	Y	DIV.D		M(A1)	Mult1	

## Registro de estado dos resultados

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

# Algoritmo de Tomasulo – CR13

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)	
L.D	F6	34+	R2	1	3	4	LD1	N
L.D	F2	45+	R3	2	4	5	LD2	N
MUL.D	F0	F2	F4	3			LD3	N
SUB.D	F8	F6	F2	4	7	8		
DIV.D	F10	F0	F6	5				
ADD.D	F6	F8	F2	6	10	11		

## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 V <sub>k</sub>	RS Qj	RS Q <sub>k</sub>
	Add1			N			
	Add2			N			
	Add3			N			
2	Mult1	Y	MUL.D	M(A2)	R(F4)		
	Mult2	Y	DIV.D		M(A1)	Mult1	

## Registro de estado dos resultados

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

# Algoritmo de Tomasulo – CR14

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)	
L.D	F6	34+	R2	1	3	4	LD1	N
L.D	F2	45+	R3	2	4	5	LD2	N
MUL.D	F0	F2	F4	3			LD3	N
SUB.D	F8	F6	F2	4	7	8		
DIV.D	F10	F0	F6	5				
ADD.D	F6	F8	F2	6	10	11		

## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1			N			
	Add2			N			
	Add3			N			
1	Mult1	Y	MUL.D	M(A2)	R(F4)		
	Mult2	Y	DIV.D		M(A1)	Mult1	

## Registro de estado dos resultados

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

# Algoritmo de Tomasulo – CR15

## Estado da instrução

Instrução	j	k	Emite final	Escreve resultado	Busy	Ender (A)
L.D	F6	34+	R2	1 3 4	LD1	N
L.D	F2	45+	R3	2 + 5	LD2	N
MUL.D	F0	F2	F4	3 15	LD3	N
SUB.D	F8	F6	F2	4 - 7 8		
DIV.D	F10	F0	F6	5		
ADD.D	F6	F8	F2	6 10 11		

## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1			N			
	Add2			N			
	Add3			N			
0	Mult1	Y	MUL.D	M(A2)	R(F4)		
	Mult2	Y	DIV.D		M(A1)	Mult1	

A multiplicação termina no fim deste CR. Quem precisa de F0?

## Registro de estado dos resultados

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

# Algoritmo de Tomasulo – CR16

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)	
L.D	F6	34+	R2	1	3	4	LD1	N
L.D	F2	45+	R3	2	4	5	LD2	N
MUL.D	F0	F2	F4	3	15	16	LD3	N
SUB.D	F8	F6	F2	4	7	8		
DIV.D	F10	F0	F6	5				
ADD.D	F6	F8	F2	6	10	11		

## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 V <sub>k</sub>	RS Qj	RS Q <sub>k</sub>
	Add1			N			
	Add2			N			
	Add3			N			
	Mult1			N			
40	Mult2	Y	DIV.D	(M*F4)	M(A1)		

## Registro de estado dos resultados

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

# Algoritmo de Tomasulo

39 ciclos de relógio mais à frente

# Algoritmo de Tomasulo – CR55

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)
L.D	F6	34+	R2	1	3	4	LD1 N
L.D	F2	45+	R3	2	4	5	LD2 N
MUL.D	F0	F2	F4	3	15	16	LD3 N
SUB.D	F8	F6	F2	4	7	8	
DIV.D	F10	F0	F6	5			
ADD.D	F6	F8	F2	6	10	11	

## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	N					
	Add2	N					
	Add3	N					
	Mult1	N					
1	Mult2	Y	DIV.D	(M*F4)	M(A1)		

## Registro de estado dos resultados

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

# Algoritmo de Tomasulo – CR56

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)	
L.D	F6	34+	R2	1	3	4	LD1	N
L.D	F2	45+	R3	2	4	5	LD2	N
MUL.D	F0	F2	F4	3	15	16	LD3	N
SUB.D	F8	F6	F2	4	7	8		
DIV.D	F10	F0	F6	5	56			
ADD.D	F6	F8	F2	6	10	11		

## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
Add1							
Add2							
Add3							
Mult1							
0	Mult2	Y	DIV.D	(M*F4)	M(A1)		

Mult2 acabou.  
Quem precisa  
de F10?

## Registro de estado dos resultados

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

# Algoritmo de Tomasulo – CR57

## Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado	Busy	Ender (A)	
L.D	F6	34+	R2	1	3	4	LD1	N
L.D	F2	45+	R3	2	4	5	LD2	N
MUL.D	F0	F2	F4	3	15	16	LD3	N
SUB.D	F8	F6	F2	4	7	o		
DIV.D	F10	F0	F6	5	56	57		
ADD.D	F6	F8	F2	6	10	11		

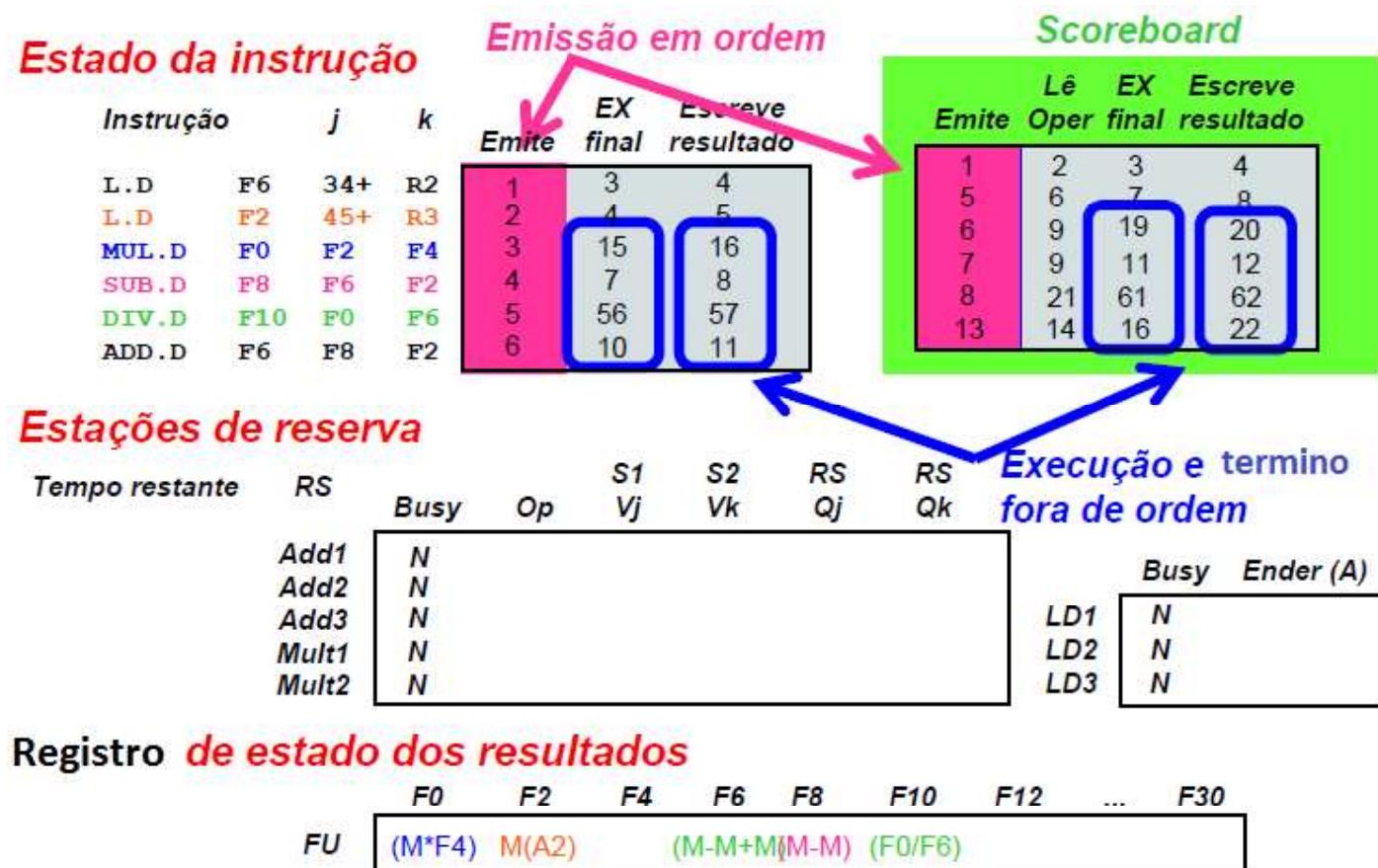
## Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
Add1							
Add2							
Add3							
Mult1							
Mult2							

## Registro de estado dos resultados

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	(M*F4)	M(A2)		(M-M+M)(M-M)	(F0/F6)				

# Algoritmo de Tomasulo – Final



# Scoreboard vs Tomasulo

## Scoreboard

**Conflitos estruturais:** bloqueia o pipeline, não emitindo instruções  
**Conflitos WAW:** bloqueia o pipeline, não emitindo instruções

**Conflitos WAR:** atrasa escrita no registrador em causa

**Forwarding:** difícil de aplicar (conflitos resolvidos no WB)

**Controlo:** centralizado em torno do Scoreboard

## Tomasulo

**Conflitos estruturais:** bloqueia o pipeline, não emitindo instruções

**Conflitos WAW:** resolvidos por renomeação nas estações de reserva

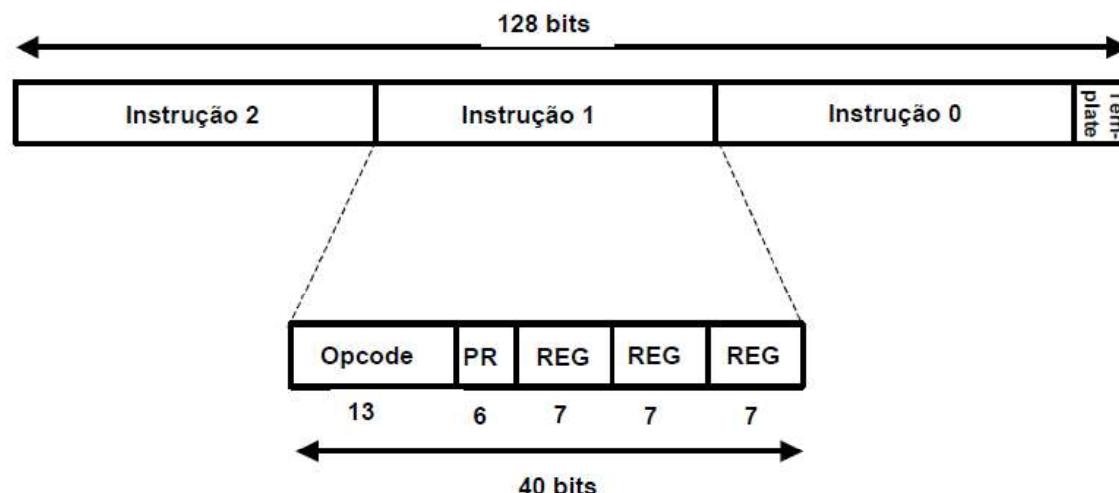
**Conflitos WAR:** resolvidos por renomeação nas estações de reserva

**Forwarding:** automaticamente clicado com o CDB

**Controlo:** distribuído pelas estações de reserva

# VLIW (Very Long Instruction Word)

- O escalonamento dinâmico **incrementa de forma considerável a complexidade do Hardware**.
- **VLIW efetua um escalonamento estático**, sendo o compilador responsável por indicar as instruções que podem ser realizadas em paralelo.
- O formato de instrução indica as operações que são realizadas em paralelo por cada unidade funcional.
- Exemplo IA-64:



## Limitações de VLIW

- O código gerado tende a ser de maior dimensão, porque é necessário inserir *nop* nos campos da instrução não preenchidos.
- Compatibilidade de código entre gerações dos mesmo processador, uma vez que tende a expor a arquitetura interna do processador
  - Compiladores diferentes são necessários
- É mais penalizado com stalls que o escalonamento dinâmico

## EPIC – IA-64 - Itanium

- 64 registos de inteiros + 60 registos FP, ambos com 64 bits
- 3 instruções com 128 bits (LIW?)
  - menos bits que VLIW clássico, produzindo código mais compacto
  - possibilidade de ligação entre os vários grupos de instruções
- Verificação de dependências em HW → compatibilidade de código

- Alguns processadores incluem unidades dedicadas à execução de operações sobre vectores de forma mais eficiente.
- Uma operação típica sobre vetores efetua uma adição de dois vetores de 64 elementos, em ponto flutuante. A operação é equivalente a uma iteração sobre os elementos do vetor, efetuando uma multiplicação por iteração

## Vantagens das operações vetoriais

- o cálculo de um resultado é independente do anterior, possibilitando a utilização de pipeline com bastante estágios, sem gerar anomalias de dados.
- uma só instrução especifica um grande número de operações, equivalente à execução de um ciclo, o que reduz a quantidade de buscas de instruções
- os acessos à memória para carregar os elementos do vetor em registradores podem tirar partido da otimização do débito da memória, amortizando o custo elevado dos acessos à memória
- as anomalias de controle são reduzidas porque os ciclos são transformados numa só instrução.

## Exemplo

$Y = \alpha X + Y$   $Y$ ,  $X$  - vetor de 64 elementos

$\alpha$  - escalar

Unidade de processamento vetorial e 8 registros de 64 valores FP

MIPS	MIPS vectorial
LD F0, a ADDI R4, Rx, 512 # último elemento Cic: LD F2, 0(Rx) # lê X(i) MULTDF2, F0, F2 # $\alpha X(i)$ LD F4, 0(Ry) # lê Y(i) ADDD F4, F2, F4 # $\alpha X(i) + Y(i)$ SD F4, 0(Ry) # guarda Y(i) ADDI Rx, Rx, 8 # inc. índice X ADDI Ry, Ry, 8 # inc. índice Y SUB R20, R4, Rx # calcula limite BEQZ R20, Cic	LD F0, a LV V1, 0(Rx) # lê vector X MULTSV V2, F0, V1 # $\alpha X$ LV V2, 0(Ry) # lê vector Y ADDV V4, V2, V3 # add SV V4, 0(Ry) # guarda Y