Tema 9. Planificación dinámica de instrucciones II: Algoritmo de Tomasulo

Organización de Computadores

LUIS ENRIQUE MORENO LORENTE
RAÚL PÉRULA MARTÍNEZ
ALBERTO BRUNETE GONZALEZ
DOMINGO MIGUEL GUINEA GARCIA ALEGRE
CESAR AUGUSTO ARISMENDI GUTIERREZ
JOSÉ CARLOS CASTILLO MONTOYA

Departamento de Ingeniería de Sistemas y Automática

REVISIÓN: SUMARIO

- Paralelismo a nivel de instrucción (ILP) en SW o HW.
- Paralelismo al nivel de bucle es más fácil de ver.
- Las dependencias del paralelismo SW definidas por programa se convierten en riesgos si el HW no puede resolverlas.
- Dependencias SW: sofisticación del compilador determina si el compilador puede desenrollar bucles.
  - Las dependencias de Memoria son las más difíciles de determinar.
- HW explotando el ILP.
  - Funciona bien cuando no se pueden conocer las dependencias en run time.
  - El código para una máquina se ejecuta bien en otras.
- Idea clave del Scoreboard: permitir a las instrucciones detrás de una burbuja que prosigan (Decode => Issue instr & read operands).
  - Permitir la ejecución out-of-order => finalización out-of-order (completion).
  - Etapa ID verifica los riesgos estructurales & dependencias de datos.

SCOREBOARD RESUMEN

- Limitaciones del scoreboard:
  - No hay hardware para forwarding.
  - Limitado a instrucciones en bloques básicos (secuencias de código sin saltos) (ventanas pequeñas).
  - Pequeño número de unidades funcionales (riesgos estructurales), especialmente unidades enteras/load store.
  - No se emite en caso de riesgo estructural.
  - Se espera por los riesgos WAR.
  - Previene los riesgos WAW.
  - No permite que dos instrucciones terminen el mismo ciclo.
  - No permite manejar unidades funcionales segmentadas.
- Mejoras
  - Una mejora fácil de introducir es permitir que el algoritmo pueda manejar unidades funcionales segmentadas.

ALGORITHMO DE TOMASULO

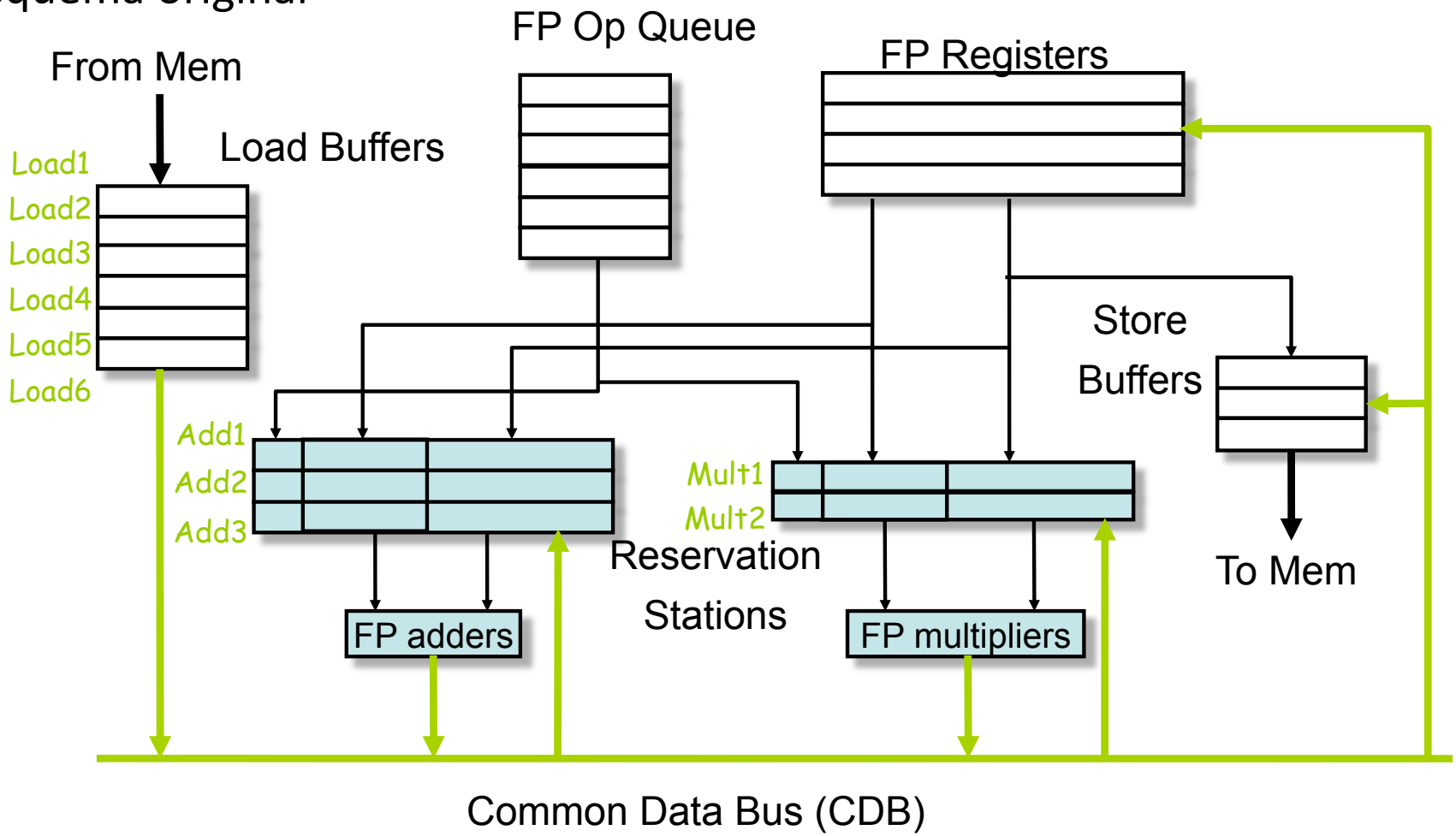
- Desarrollado para el IBM 360/91 3 años después del CDC 6600 (1966)
- **Objetivo**: Obtener altas prestaciones sin utilizar compiladores especiales
- Diferencias entre IBM 360 & CDC 6600 ISA
  - IBM tiene sólo 2 especificadores de registro/instr vs. 3 en el CDC 6600
  - IBM tiene 4 FP registros vs. 8 en CDC 6600
- ¿Por qué estudiarlos? Conducen al Alpha 21264, HP 8000, MIPS 10000, Pentium II, PowerPC 604, …

TOMASULO VS. SCOREBOARD

- Control & buffers distribuidos con Unidades Funcionales(FU) en vez del control y buffers centralizado utilizado en scoreboard;
  - Los buffers de las FU se denominan "estaciones de reserva"; contienen los operandos pendientes.
- Los Registros en las instrucciones son reemplazados por valores o punteros a las estaciones de reserva(RS); esto se denomina renombrado de registros;
  - Evita los riesgos WAR, WAW.
  - Hay más estaciones de reserva que registros, por lo que pueden realizar optimizaciones que los compiladores no pueden.
- Los resultados de las FU van a las RS, pero no a través de los registros, el Common Data Bus es el que difunde los resultados a todas las FUs.
- Load y Stores, se tratan como FUs con RSs también.
- Instrucciones Enteras pueden adelantar a los branches, permitiendo las operaciones FP más allá de los bloques básicos en la cola de FP.

TOMASULO: ORGANIZACIÓN

Esquema original



COMPONENTES DE LAS ESTACIONES DE RESERVA

Op—Operación a realizar en la unidad (e.g., + or –)

Vj, Vk—Valores de los operandos fuente

- Buffers de Store tienen un campo V , resultado a ser almacenado

Qj, Qk—Estaciones de Reserva producen los registros fuente (el valor a escribir)

- Nota: No hay flags de ready como en Scoreboard; Qj,Qk=0 => ready
- Los buffers de Store sólo tienen Qi para la estación de reserva RS que produce el resultado

Busy—Indica si la estación de reserva or FU esta ocupada

Register result status—Indica qué unidad funcional escribirá en cada registro, si es que hay alguno. En blanco cuando no hay ninguna instrucción pendiente que vaya a escribir ese registro.

ETAPAS DEL ALGORITMO DE TOMASULO

1. Issue—obtiene la instrucción de la cola de Op (si no hay cola del buffer)

   Si la estación de reserva está libre (no hay riesgo estructural),
   el control emite la instr & envía los operandos (renombrando los registros).

2. Ejecución—opera sobre los operandos (EX)

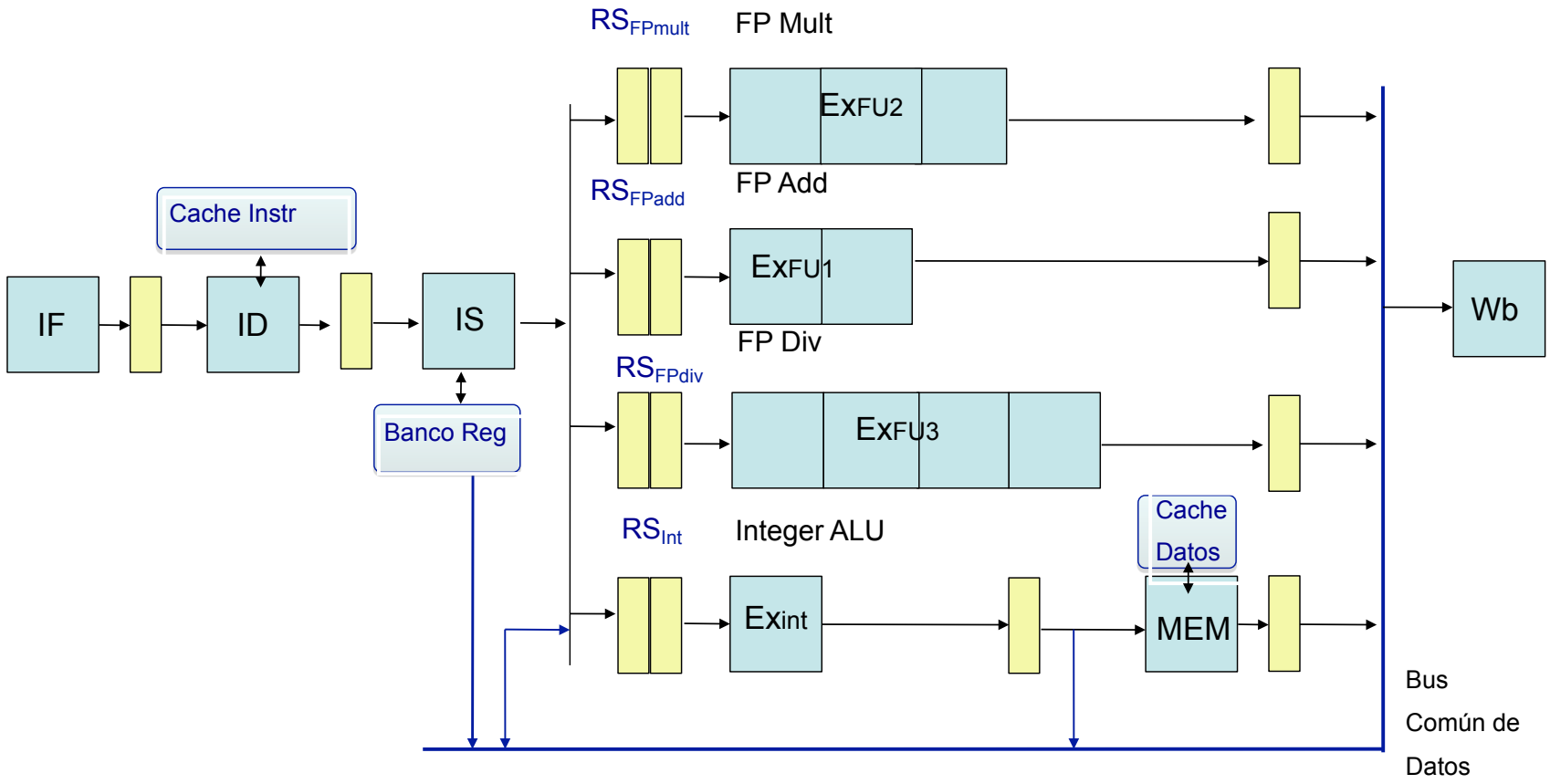
   Cuando ambos operandos están disponibles entonces se ejecuta;
   Si no están disponibles, se observa el Common Data Bus por el resultado

3. Escritura del resultado—finaliza la ejecución (WB)

   Escribe en el Common Data Bus para todas las unidades que esperan; marca la estación de reserva como disponible

- Normal data bus: data + destino ("go to" bus)
- Common data bus: data + fuente ("come from" bus)
  - 64 bits de datos + 4 bits de dirección fuente de la U.Funcional
  - Escribe si casa con la U. Funcional esperada (produce el resultado)
  - Realiza el broadcast

CAUCE DLX CON TOMASULO



RESUMEN ALGORITMO

Instruction state	Wait until	Action or bookkeeping

Issue
FP operation
	Station r empty	`if (RegisterStat[rs].Qi≠0)`
		`  {RS[r].Qj ← RegisterStat[rs].Qi}`
		`else {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0};`
		`if (RegisterStat[rt].Qi≠0)`
		`  {RS[r].Qk ← RegisterStat[rt].Qi`
		`else {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0};`
		`RS[r].Busy ← yes; RegisterStat[rd].Q ← r;`

Load or store	Buffer r empty	`if (RegisterStat[rs].Qi≠0)`
		`  {RS[r].Qj ← RegisterStat[rs].Qi}`
		`else {RS[r].Vj ← Regs[rs]; RS[r].Qj ← 0};`
		`RS[r].A ← imm; RS[r].Busy ← yes;`

Load only		`RegisterStat[rt].Qi ← r;`

Store only		`if (RegisterStat[rt].Qi≠0)`
		`  {RS[r].Qk ← RegisterStat[rs].Qi}`
		`  else {RS[r].Vk ← Regs[rt]; RS[r].Qk ← 0};`

Execute FP operation	(RS[r].Qj = 0) and (RS[r].Qk = 0)	Compute result: operands are in Vj and Vk
Load-store step 1	RS[r].Qj = 0 & r is head of load-store queue	`RS[r].A ← RS[r].Vj + RS[r].A;`
Load step 2	Load step 1 complete	Read from Mem[RS[r].A]

Write Result
FP operation or load
	Execution complete at r & CDB available	`∀x(if (RegisterStat[x].Qi=r) {Regs[x] ← result;`
		`  RegisterStat[x].Qi ← 0});`
		`∀x(if (RS[x].Qj=r) {RS[x].Vj ← result;RS[x].Qj ←`
		`  0});`
		`∀x(if (RS[x].Qk=r) {RS[x].Vk ← result;RS[x].Qk ←`
		`  0});`
		`RS[r].Busy ← no;`

Store	Execution complete at r & RS[r].Qk = 0	`Mem[RS[r].A] ← RS[r].Vk;` `RS[r].Busy ← no;`

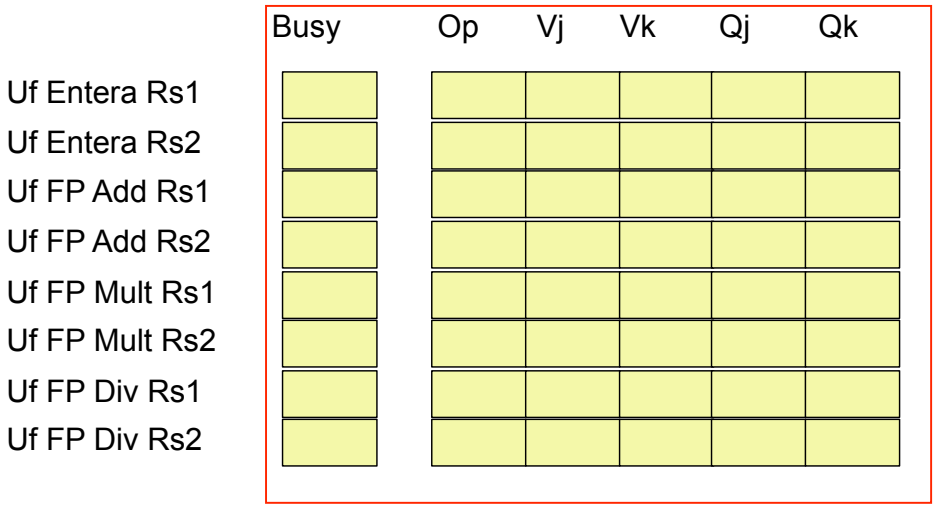
*Image from J. L. Hennesy and D. Patterson "Computer Arquitecture: A Quantitative Approach"

ESTRUCTURA DE DATOS BÁSICA PARA DLX

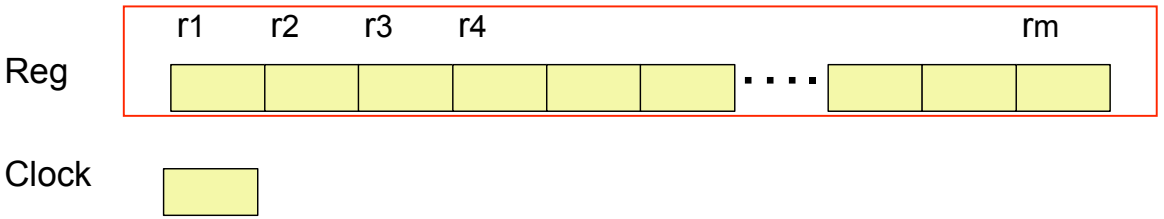
Instr. status

Instr.	If	Id	Is	Ex	Wb
i1					
i2					
i3					
i4					

Functional Unit status



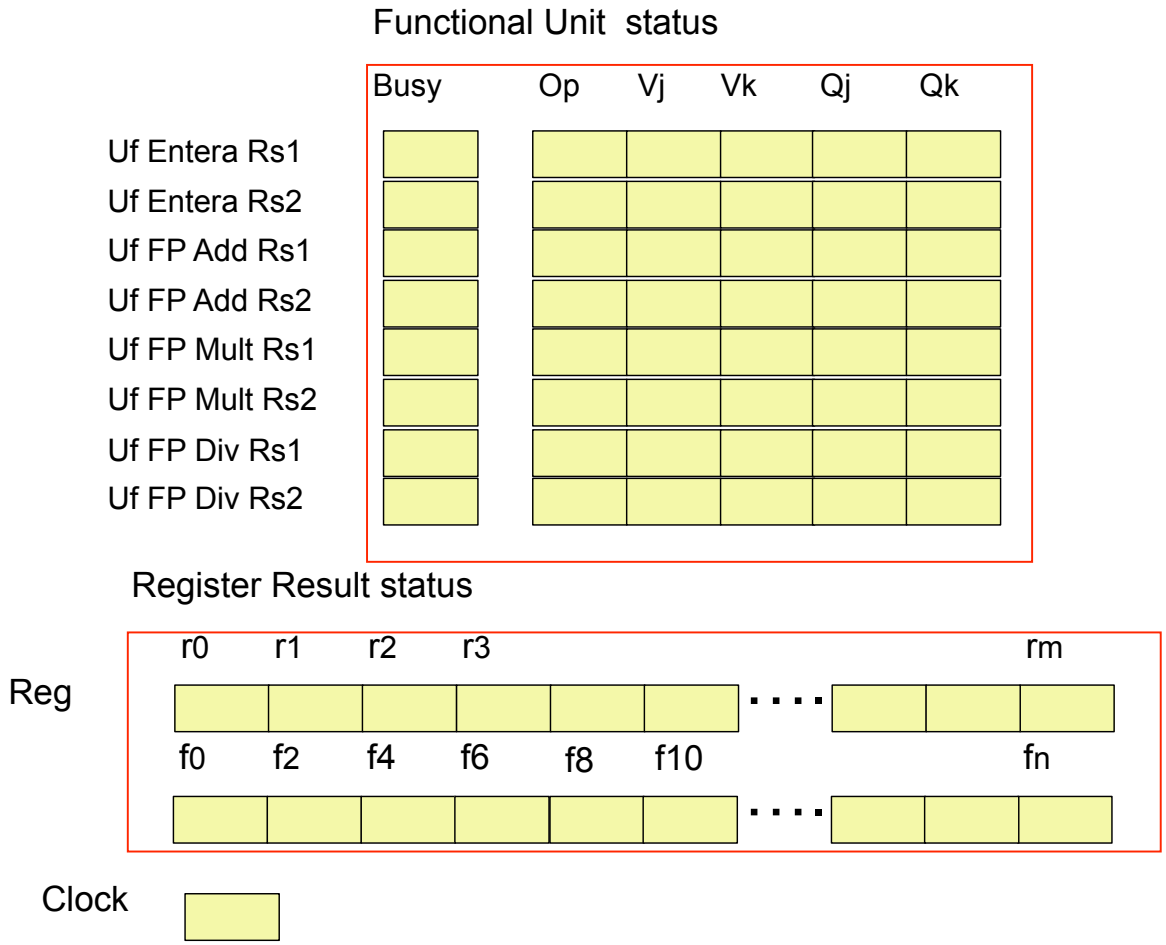
Register Result status



DLX CON TOMASULO: EJEMPLO

Instr. status

Instr.	If	Id	Is	Ex	W
ld f6,34+,r2					
ld f2,45+,r3					
multd f0,f2,f4					
subd f8,f6,f2					
divd f10,f0,f6					
addd f6,f8,f2					



DLX CON TOMASULO: EJEMPLO → CICLO 1

Instr. status

Instr.	If	Id	Is	Ex	W
ld f6,34+,r2	1				
ld f2,45+,r3					
multd f0,f2,f4					
subd f8,f6,f2					
divd f10,f0,f6					
addd f6,f8,f2					

Functional Unit status

	Busy	Op	Vj	Vk	Qj	Qk
Uf Entera Rs1	No					
Uf Entera Rs2	No					
Uf FP Add Rs1	No					
Uf FP Add Rs2	No					
Uf FP Mult Rs1	No					
Uf FP Mult Rs2	No					
Uf FP Div Rs1	No					
Uf FP Div Rs2	No					

Register Result status

Reg

r0	r1	r2	r3	. . . .	rm
f0	f2	f4	f6	f8 f10	fn

Clock 1

EJEMPLO → CRONOGRAMA (CICLO 1)

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld f6,34+,r2	If																			
ld f2,45+,r3																				
multd f0,f2,f4																				
subd f8,f6,f2																				
divd f10,f0,f6																				
addd f6,f8,f2																				

DLX CON TOMASULO: EJEMPLO → CICLO 2

Instr. status

Instr.	If	Id	Is	Ex	W
ld f6,34+,r2	1	2			
ld f2,45+,r3	2				
multd f0,f2,f4					
subd f8,f6,f2					
divd f10,f0,f6					
addd f6,f8,f2					

Functional Unit status

	Busy	Op	Vj	Vk	Qj	Qk
Uf Entera Rs1	No					
Uf Entera Rs2	No					
Uf FP Add Rs1	No					
Uf FP Add Rs2	No					
Uf FP Mult Rs1	No					
Uf FP Mult Rs2	No					
Uf FP Div Rs1	No					
Uf FP Div Rs2	No					

Register Result status

Reg

r0	r1	r2	r3	. . . .	rm
f0	f2	f4	f6	f8 f10	fn

Clock 2

EJEMPLO → CRONOGRAMA (CICLO 2)

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld f6,34+,r2	If	Id																		
ld f2,45+,r3		If																		
multd f0,f2,f4																				
subd f8,f6,f2																				
divd f10,f0,f6																				
addd f6,f8,f2																				

DLX CON TOMASULO: EJEMPLO → CICLO 3

Instr. status

Instr.	If	Id	Is	Ex	W
ld f6,34+,r2	1	2	3		
ld f2,45+,r3	2	3			
multd f0,f2,f4	3				
subd f8,f6,f2					
divd f10,f0,f6					
addd f6,f8,f2					

Functional Unit status

	Busy	Op	Vj	Vk	Qj	Qk
Uf Entera Rs1	Yes	2 Ld	+34	r2		
Uf Entera Rs2	No					
Uf FP Add Rs1	No					
Uf FP Add Rs2	No					
Uf FP Mult Rs1	No					
Uf FP Mult Rs2	No					
Uf FP Div Rs1	No					
Uf FP Div Rs2	No					

Register Result status

Reg

r0	r1	r2	r3	. . . .	rm
f0	f2	f4	f6	f8 f10	fn
			EnRs1		

Clock 3

EJEMPLO → CRONOGRAMA (CICLO 3)

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld f6,34+,r2	If	Id	Is																	
ld f2,45+,r3		If	Id																	
multd f0,f2,f4			If																	
subd f8,f6,f2																				
divd f10,f0,f6																				
addd f6,f8,f2																				

DLX CON TOMASULO: EJEMPLO → CICLO 4

Instr. status

Instr.	If	Id	Is	Ex	W
ld f6,34+,r2	1	2	3	4	
ld f2,45+,r3	2	3	4		
multd f0,f2,f4	3	4			
subd f8,f6,f2	4				
divd f10,f0,f6					
addd f6,f8,f2					

Functional Unit status

	Busy		Op	Vj	Vk	Qj	Qk
Uf Entera Rs1	Yes	1	Ld	+34	r2		
Uf Entera Rs2	Yes	2	Ld	+45	r3		
Uf FP Add Rs1	No						
Uf FP Add Rs2	No						
Uf FP Mult Rs1	No						
Uf FP Mult Rs2	No						
Uf FP Div Rs1	No						
Uf FP Div Rs2	No						

Register Result status

Reg	r0	r1	r2	r3	. . . .	rm
	f0	f2	f4	f6	f8 f10	fn
		EnRs2		EnRs1		

Clock 4

EJEMPLO → CRONOGRAMA (CICLO 4)

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld f6,34+,r2	If	Id	Is	Ex																
ld f2,45+,r3		If	Id	Is																
multd f0,f2,f4			If	Id																
subd f8,f6,f2				If																
divd f10,f0,f6																				
addd f6,f8,f2																				

DLX CON TOMASULO: EJEMPLO → CICLO 5

Instr. status

Instr.	If	Id	Is	Ex	W
ld f6,34+,r2	1	2	3	5	
ld f2,45+,r3	2	3	4	5	
multd f0,f2,f4	3	4	5		
subd f8,f6,f2	4	5			
divd f10,f0,f6	5				
addd f6,f8,f2					

Functional Unit status

	Busy		Op	Vj	Vk	Qj	Qk
Uf Entera Rs1	Yes	0	Ld	+34	r2		
Uf Entera Rs2	Yes	1	Ld	+45	r3		
Uf FP Add Rs1	No						
Uf FP Add Rs2	No						
Uf FP Mult Rs1	Yes	3o	Mul		f4	EnRs2	
Uf FP Mult Rs2	No						
Uf FP Div Rs1	No						
Uf FP Div Rs2	No						

Register Result status

Reg	r0	r1	r2	r3				rm
	f0	f2	f4	f6	f8	f10		fn
	MuRs1	EnRs2		EnRs1				

Clock 5

EJEMPLO → CRONOGRAMA (CICLO 5)

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld f6,34+,r2	If	Id	Is	Ex	M															
ld f2,45+,r3		If	Id	Is	Ex															
multd f0,f2,f4			If	Id	Is															
subd f8,f6,f2				If	Id															
divd f10,f0,f6					If															
addd f6,f8,f2																				

DLX CON TOMASULO: EJEMPLO → CICLO 6

Instr. status

Instr.	If	Id	Is	Ex	W
ld f6,34+,r2	1	2	3	5	6
ld f2,45+,r3	2	3	4	6	
multd f0,f2,f4	3	4	5		
subd f8,f6,f2	4	5	6		
divd f10,f0,f6	5	6			
addd f6,f8,f2	6				

Functional Unit status

	Busy		Op	Vj	Vk	Qj	Qk
Uf Entera Rs1	No						
Uf Entera Rs2	Yes	0	Ld	+45	r3		
Uf FP Add Rs1	Yes	2o	Sub	f6			EnRs2
Uf FP Add Rs2	No						
Uf FP Mult Rs1	Yes	3o	Mul		f4	EnRs2	
Uf FP Mult Rs2	No						
Uf FP Div Rs1	No						
Uf FP Div Rs2	No						

Register Result status

Reg	r0	r1	r2	r3				rm
	f0	f2	f4	f6	f8	f10		fn
	MuRs1	EnRs2			AdRs2			

Clock 6

EJEMPLO → CRONOGRAMA (CICLO 6)

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld f6,34+,r2	If	Id	Is	Ex	M	W														
ld f2,45+,r3		If	Id	Is	Ex	M														
multd f0,f2,f4			If	Id	Is	o														
subd f8,f6,f2				If	Id	Is														
divd f10,f0,f6					If	Id														
addd f6,f8,f2						If														

DLX CON TOMASULO: EJEMPLO → CICLO 7

Instr. status

Instr.	If	Id	Is	Ex	W
ld f6,34+,r2	1	2	3	5	6
ld f2,45+,r3	2	3	4	6	7
multd f0,f2,f4	3	4	5	7	
subd f8,f6,f2	4	5	6	7	
divd f10,f0,f6	5	6	7		
addd f6,f8,f2	6	7			

Functional Unit status

	Busy		Op	Vj	Vk	Qj	Qk
Uf Entera Rs1	No						
Uf Entera Rs2	Yes	0	Ld	+45	r3		
Uf FP Add Rs1	Yes	1	Sub	f6	f2		
Uf FP Add Rs2	No						
Uf FP Mult Rs1	Yes	2	Mul	f2	f4		
Uf FP Mult Rs2	No						
Uf FP Div Rs1	Yes	4o	Div		f6	MuRs1	
Uf FP Div Rs2	No						

Register Result status

Reg	r0	r1	r2	r3				rm
	f0	f2	f4	f6	f8	f10		fn
	MuRs1				AdRs1	DvRs1		

Clock 7

EJEMPLO → CRONOGRAMA (CICLO 7)

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld f6,34+,r2	If	Id	Is	Ex	M	W														
ld f2,45+,r3		If	Id	Is	Ex	M	W													
multd f0,f2,f4			If	Id	Is	o	Ex_m													
subd f8,f6,f2				If	Id	Is	Ex_a													
divd f10,f0,f6					If	Id	Is													
addd f6,f8,f2						If	Id													

DLX CON TOMASULO: EJEMPLO → CICLO 8

Instr. status

Instr.	If	Id	Is	Ex	W
ld f6,34+,r2	1	2	3	5	6
ld f2,45+,r3	2	3	4	6	7
multd f0,f2,f4	3	4	5	8	
subd f8,f6,f2	4	5	6	8	
divd f10,f0,f6	5	6	7		
addd f6,f8,f2	6	7	8		

Functional Unit status

	Busy		Op	Vj	Vk	Qj	Qk
Uf Entera Rs1	No						
Uf Entera Rs2	No						
Uf FP Add Rs1	Yes	0	Sub	f6	f2		
Uf FP Add Rs2	Yes	2o	Add	f2			AdRs1
Uf FP Mult Rs1	Yes	1	Mul	f2	f4		
Uf FP Mult Rs2	No						
Uf FP Div Rs1	Yes	4o	Div		f6	MuRs1	
Uf FP Div Rs2	No						

Register Result status

Reg

r0	r1	r2	r3	. . . .	rm
f0	f2	f4	f6	f8	f10	. . . .	fn
MuRs1			AdRs2	AdRs1	DvRs1	. . . .	

Clock 8

EJEMPLO → CRONOGRAMA (CICLO 8)

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld f6,34+,r2	If	Id	Is	Ex_i	M	W														
ld f2,45+,r3		If	Id	Is	Ex_i	M	W													
multd f0,f2,f4			If	Id	Is	o	Ex_m	Ex_m												
subd f8,f6,f2				If	Id	Is	Ex_a	Ex_a												
divd f10,f0,f6					If	Id	Is	o												
addd f6,f8,f2						If	Id	Is												

DLX CON TOMASULO: EJEMPLO → CICLO 9

Instr. status

Instr.	If	Id	Is	Ex	W
ld f6,34+,r2	1	2	3	5	6
ld f2,45+,r3	2	3	4	6	7
multd f0,f2,f4	3	4	5	9	
subd f8,f6,f2	4	5	6	8	9
divd f10,f0,f6	5	6	7		
addd f6,f8,f2	6	7	8	9	

Functional Unit status

	Busy		Op	Vj	Vk	Qj	Qk
Uf Entera Rs1	No						
Uf Entera Rs2	No						
Uf FP Add Rs1	No						
Uf FP Add Rs2	Yes	1	Add	f2	f8		
Uf FP Mult Rs1	Yes	0	Mul	f2	f4		
Uf FP Mult Rs2	No						
Uf FP Div Rs1	Yes	4o	Div		f6	MuRs1	
Uf FP Div Rs2	No						

Register Result status

Reg	r0	r1	r2	r3				rm
	f0	f2	f4	f6	f8	f10		fn
	MuRs1			AdRs2		DvRs1		

Clock 9

EJEMPLO → CRONOGRAMA (CICLO 9)

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld f6,34+,r2	If	Id	Is	Ex_i	M	W														
ld f2,45+,r3		If	Id	Is	Ex_i	M	W													
multd f0,f2,f4			If	Id	Is	o	Ex_m	Ex_m	Ex_m											
subd f8,f6,f2				If	Id	Is	Ex_a	Ex_a	W											
divd f10,f0,f6					If	Id	Is	o	o											
addd f6,f8,f2						If	Id	Is	Ex_d											

DLX CON TOMASULO: EJEMPLO → CICLO 10

Instr. status

Instr.	If	Id	Is	Ex	W
ld f6,34+,r2	1	2	3	5	6
ld f2,45+,r3	2	3	4	6	7
multd f0,f2,f4	3	4	5	9	10
subd f8,f6,f2	4	5	6	8	9
divd f10,f0,f6	5	6	7	10	
addd f6,f8,f2	6	7	8	10	

f4

Functional Unit status

	Busy		Op	Vj	Vk	Qj	Qk
Uf Entera Rs1	No						
Uf Entera Rs2	No						
Uf FP Add Rs1	No						
Uf FP Add Rs2	Yes	0	Add	f2	f8		
Uf FP Mult Rs1	No						
Uf FP Mult Rs2	No						
Uf FP Div Rs1	Yes	3	Div	f0	f6		
Uf FP Div Rs2	No						

Register Result status

Reg	r0	r1	r2	r3				rm
	f0	f2	f4	f6	f8	f10		fn
				AdRs2		DvRs1		

Clock 10

EJEMPLO → CRONOGRAMA (CICLO 10)

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld f6,34+,r2	If	Id	Is	Ex_i	M	W														
ld f2,45+,r3		If	Id	Is	Ex_i	M	W													
multd f0,f2,f4			If	Id	Is	o	Ex_m	Ex_m	Ex_m	W										
subd f8,f6,f2				If	Id	Is	Ex_a	Ex_a	W											
divd f10,f0,f6					If	Id	Is	o	o	Ex_d										
addd f6,f8,f2						If	Id	Is	Ex_a	Ex_a										

DLX CON TOMASULO: EJEMPLO → CICLO 11

Instr. status

Instr.	If	Id	Is	Ex	W
ld f6,34+,r2	1	2	3	5	6
ld f2,45+,r3	2	3	4	6	7
multd f0,f2,f4	3	4	5	9	10
subd f8,f6,f2	4	5	6	8	9
divd f10,f0,f6	5	6	7	11	
addd f6,f8,f2	6	7	8	10	11

Functional Unit status

	Busy	Op	Vj	Vk	Qj	Qk
Uf Entera Rs1	No					
Uf Entera Rs2	No					
Uf FP Add Rs1	No					
Uf FP Add Rs2	No					
Uf FP Mult Rs1	No					
Uf FP Mult Rs2	No					
Uf FP Div Rs1	Yes	2 Div	f0	f6		
Uf FP Div Rs2	No					

Register Result status

Reg	r0	r1	r2	r3	. . . .	rm
	f0	f2	f4	f6	f8	f10	. . . .	fn
						DvRs1		

Clock 11

EJEMPLO → CRONOGRAMA (CICLO 11)

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld f6,34+,r2	If	Id	Is	Ex_i	M	W														
ld f2,45+,r3		If	Id	Is	Ex_i	M	W													
multd f0,f2,f4			If	Id	Is	o	Ex_m	Ex_m	Ex_m	W										
subd f8,f6,f2				If	Id	Is	Ex_a	Ex_a	W											
divd f10,f0,f6					If	Id	Is	o	o	Ex_d	Ex_d									
addd f6,f8,f2						If	Id	Is	Ex_a	Ex_a	W									

DLX CON TOMASULO: EJEMPLO → CICLO 12

Instr. status

Instr.	If	Id	Is	Ex	W
ld f6,34+,r2	1	2	3	5	6
ld f2,45+,r3	2	3	4	6	7
multd f0,f2,f4	3	4	5	9	10
subd f8,f6,f2	4	5	6	8	9
divd f10,f0,f6	5	6	7	12	
addd f6,f8,f2	6	7	8	10	11

Functional Unit status

	Busy	Op	Vj	Vk	Qj	Qk
Uf Entera Rs1	No					
Uf Entera Rs2	No					
Uf FP Add Rs1	No					
Uf FP Add Rs2	No					
Uf FP Mult Rs1	No					
Uf FP Mult Rs2	No					
Uf FP Div Rs1	Yes	Div	f0	f6		
Uf FP Div Rs2	No					

1

Register Result status

Reg	r0	r1	r2	r3	. . . .	rm
	f0	f2	f4	f6	f8	f10	. . . .	fn
						DvRs1		

Clock 11

EJEMPLO → CRONOGRAMA (CICLO 11)

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld f6,34+,r2	If	Id	Is	Ex_i	M	W														
ld f2,45+,r3		If	Id	Is	Ex_i	M	W													
multd f0,f2,f4			If	Id	Is	o	Ex_m	Ex_m	Ex_m	W										
subd f8,f6,f2				If	Id	Is	Ex_a	Ex_a	W											
divd f10,f0,f6					If	Id	Is	o	o	Ex_d	Ex_d	Ex_d								
addd f6,f8,f2						If	Id	Is	Ex_a	Ex_a	W									

DLX CON TOMASULO: EJEMPLO → CICLO 13

Instr. status

Instr.	If	Id	Is	Ex	W
ld f6,34+,r2	1	2	3	5	6
ld f2,45+,r3	2	3	4	6	7
multd f0,f2,f4	3	4	5	9	10
subd f8,f6,f2	4	5	6	8	9
divd f10,f0,f6	5	6	7	13	
addd f6,f8,f2	6	7	8	10	11

Functional Unit status

	Busy	Op	Vj	Vk	Qj	Qk
Uf Entera Rs1	No					
Uf Entera Rs2	No					
Uf FP Add Rs1	No					
Uf FP Add Rs2	No					
Uf FP Mult Rs1	No					
Uf FP Mult Rs2	No					
Uf FP Div Rs1	Yes	Div	f0	f6		
Uf FP Div Rs2	No					

0

Register Result status

Reg

r0	r1	r2	r3	...	rm
f0	f2	f4	f6	f8	f10	...	fn
					DvRs1		

Clock 13

EJEMPLO → CRONOGRAMA (CICLO 13)

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld f6,34+,r2	If	Id	Is	Ex_i	M	W														
ld f2,45+,r3		If	Id	Is	Ex_i	M	W													
multd f0,f2,f4			If	Id	Is	o	Ex_m	Ex_m	Ex_m	W										
subd f8,f6,f2				If	Id	Is	Ex_a	Ex_a	W											
divd f10,f0,f6					If	Id	Is	o	o	Ex_d	Ex_d	Ex_d	Ex_d							
addd f6,f8,f2						If	Id	Is	Ex_a	Ex_a	W									

DLX CON TOMASULO: EJEMPLO → CICLO 14

Instr. status

Instr.	If	Id	Is	Ex	W
ld f6,34+,r2	1	2	3	5	6
ld f2,45+,r3	2	3	4	6	7
multd f0,f2,f4	3	4	5	9	10
subd f8,f6,f2	4	5	6	8	9
divd f10,f0,f6	5	6	7	13	14
addd f6,f8,f2	6	7	8	10	11

Functional Unit status

f4

	Busy	Op	Vj	Vk	Qj	Qk
Uf Entera Rs1	No					
Uf Entera Rs2	No					
Uf FP Add Rs1	No					
Uf FP Add Rs2	No					
Uf FP Mult Rs1	No					
Uf FP Mult Rs2	No					
Uf FP Div Rs1	No					
Uf FP Div Rs2	No					

Register Result status

Reg

r0	r1	r2	r3	. . . .	rm
f0	f2	f4	f6	f8 f10	fn

Clock 14

EJEMPLO → CRONOGRAMA (CICLO 14)

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld f6,34+,r2	If	Id	Is	Ex_i	M	W														
ld f2,45+,r3		If	Id	Is	Ex_i	M	W													
multd f0,f2,f4			If	Id	Is	o	Ex_m	Ex_m	Ex_m	W										
subd f8,f6,f2				If	Id	Is	Ex_a	Ex_a	W											
divd f10,f0,f6					If	Id	Is	o	o	Ex_d	Ex_d	Ex_d	Ex_d	W						
addd f6,f8,f2						If	Id	Is	Ex_a	Ex_a	W									

TOMASULO VS. SCOREBOARD (IBM 360/91 V. CDC 6600)

Unidades funcionales segmentadas	Multiples Unidades Funcionales
(6 load, 3 store, 3 +, 2 x/÷)	(1 load/store, 1 + , 2 x, 1 ÷)
tamaño de ventana: 14 instrucciones	5 instrucciones
No se emite si existe riesgo estructural	igual
WAR: el renombrado los evita	detiene la finalización
WAW: el renombrado los evita	detiene la finalización
Los resultados se difunden desde las UF	Registros de Write/read
Control: estaciones de reserva (distribuido)	scoreboard central

TOMASULO: INCONVENIENTES

- Complejidad
  - Retardos de 360/91, MIPS 10000, Alpha 21264, IBM PPC 620.
- Muchas cargas de registros asociativos por ciclo.
- Prestaciones limitadas por el Common Data Bus.
  - Cada CDB debe ir a múltiples unidades funcionales.
    - Alta capacitancia, alta densidad de cableado.
  - El número de unidades funcionales que pueden ser completados en un ciclo está limitado a una.
    - Múltiples CDBs => más lógica en UF para los stores asociativos.
- Interrupciones no-precisas.

EL ALGORITMO DE TOMASULO OFRECE DOS GRANDES VENTAJAS

(1) La lógica de detección de riesgos está distribuida

- Estaciones de reserva distribuidas y bus común de datos (CDB)
- Si múltiples instrucciones esperan por un único resultado, y cada instrucción tiene otro operando, entonces las instrucciones pueden ser activadas simultáneamente emitiendo este resultado por el bus común de datos
- Si se usase un banco de registros centralizado, las unidades tendrían que leer los resultados del banco de registros cuando los buses estuviesen disponibles

(2) La eliminación de las detenciones por riesgos WAW y WAR

- Renombrado de registros en las estaciones de reserva
- Almacenar operandos en las estaciones de reserva tan pronto como están disponibles.

TOMASULO: SUMARIO

- Estaciones de reserva: renombrado en un conjunto mayor de registros + buffers de operandos fuente.
  - Previene que los registros sean cuellos de botella.
  - Evita los riesgos WAR, WAW del Scoreboard.
  - Permite desenrollado de bucles en in HW.
- No está limitado a bloques básicos (secuencia de código sin saltos)
  (unidades enteras se adelantan, más allá de los branches).
- Ayuda en las “cache misses” también .
- Últimas contribuciones.
  - Planificación dinámica.
  - Renombrado de registros.
  - Desambiguación Load/store.
- 360/91 descendientes: Pentium II; PowerPC 604; MIPS R10000; HP-PA 8000; Alpha 21264.