Tema 8. Introducción a la planificación dinámica de instrucciones: Scoreboard

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

PLANIFICACIÓN DINÁMICA: VENTAJAS

- Maneja aquellos casos en los que las dependencias son desconocidas durante la compilación.
  - (ej., porque pueden involucrar referencias a memoria).
- Simplifica el compilador.
- Permite que el código que ha sido compilado para un cierto procesador segmentado pueda ser ejecutado eficientemente en otro procesador segmentado diferente.
- La especulación hardware, una técnica que permite ventajas de eficiencia significativas, requiere de planificación dinámica.

ESQUEMAS HW: PARALELISMO DE INSTRUCCIONES

- Idea clave: permitir que las instrucciones posteriores a una detenida puedan comenzar a ejecutarse.

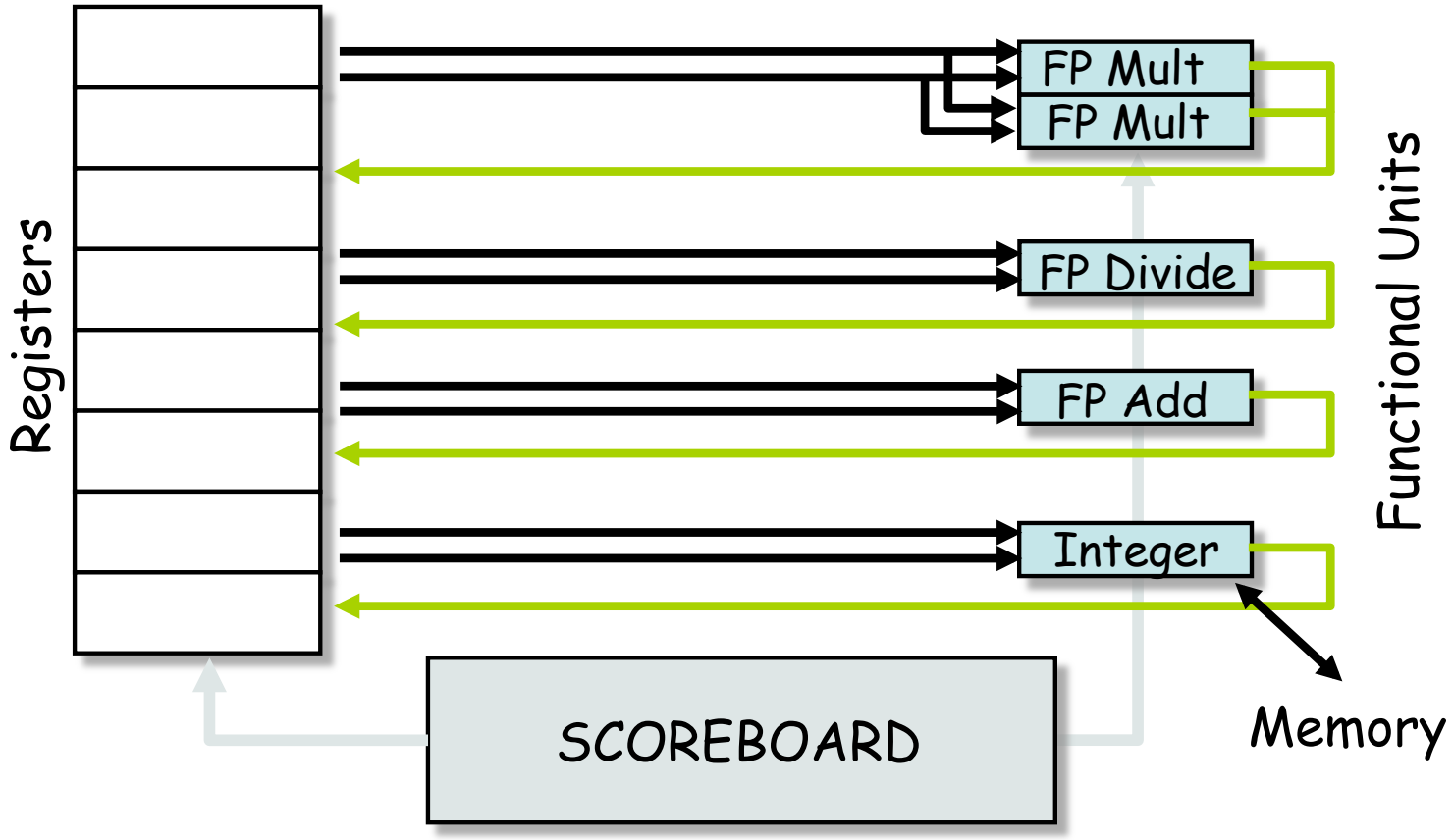
```
DIVD F0,F2,F4
ADDD F10,F0,F8
SUBD F12,F8,F14
```

  - Permite la ejecución out-of-order => finalización out-of-order (completion).
  - Permite distinguir entre el comienzo de la ejecución y la finalización de la ejecución de una instrucción; entre esos dos instantes de tiempo la instrucción está en ejecución.
  - En un cauce con planificación dinámica, todas las instrucciones pasan por una etapa de iniciación en orden (in-order).
  - La etapa ID descodifica y comprueba los riesgos estructurales. Scoreboard se utilizó por vez primera en la CDC 6600 en 1963.

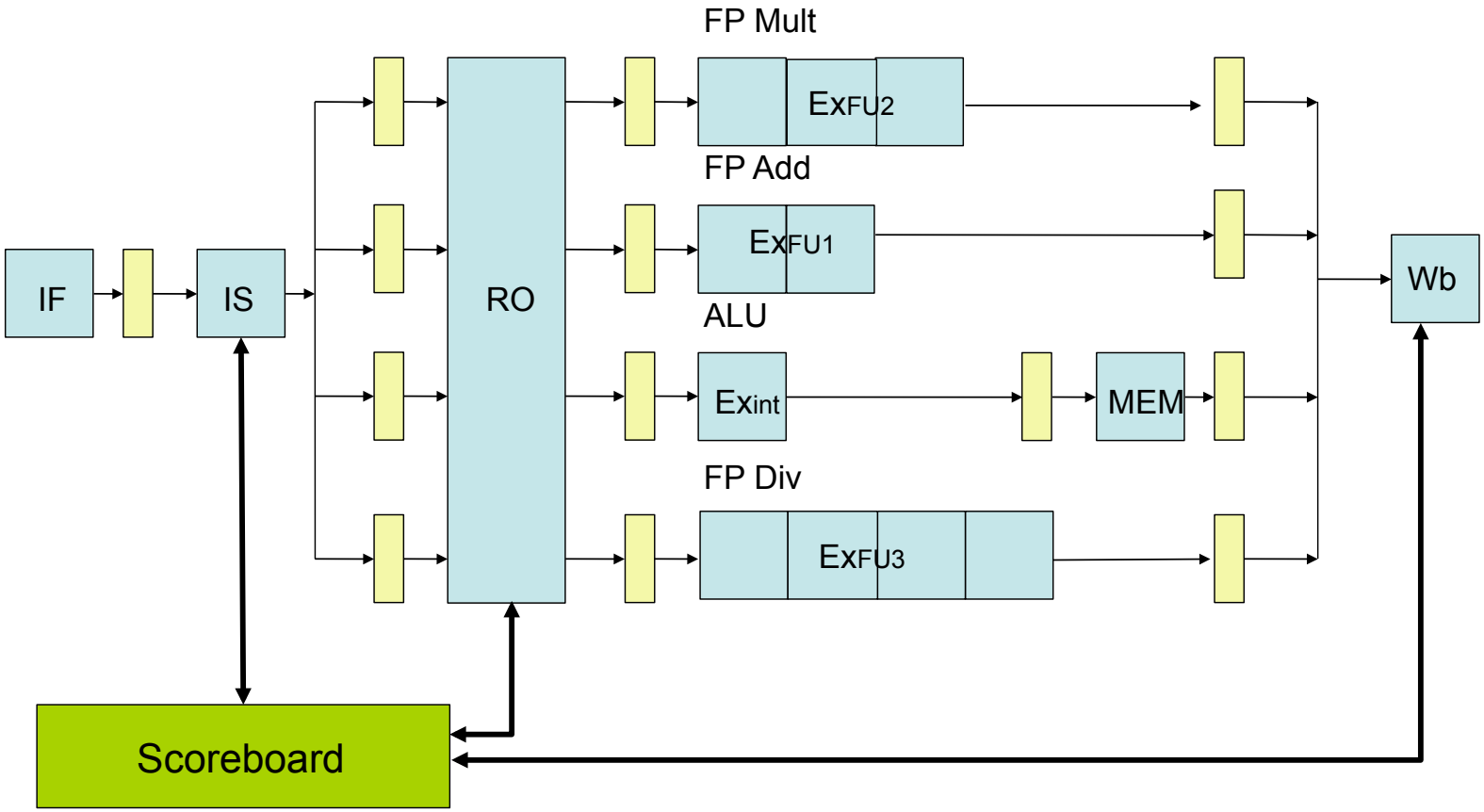
ESQUEMAS HW: PARALELISMO DE INSTRUCCIONES

- Un cauce simple tiene una etapa para verificar tanto los riesgos estructurales como los de datos: la etapa Instruction Decode (ID), también denominada Instruction Issue.
- La ejecución Out-of-order divide la etapa ID en dos:
  1. Issue (Emisión)—descodifica las instrucciones, comprueba los riesgos estructurales.
  2. Lectura de operandos—espera hasta que no existan riesgos de datos, entonces lee los operandos.
- Scoreboards permite a la instrucción ejecutarse cuando 1 & 2 se han realizado, y no se espera por instrucciones previas.
- CDC 6600: emite las instr. in order, las ejecuta out of order , y las completa out of order (commit o completion en inglés).

ARQUITECTURA ORIGINAL DEL SCOREBOARD (CDC 6600)



CAUCE DLX CON SCOREBOARD



SCOREBOARD: IMPLICACIONES

- Finalización Out-of-order => riesgos WAR, WAW?
- Soluciones para los riesgos WAR.
  - Encola ambas operaciones y copia de sus operandos.
  - Lee los registros solamente durante la etapa de lectura de operandos.
- Los riesgos WAW, deben detectarse y detener hasta que otras instrucciones se completen.
- Se necesita tener múltiples instrucciones en la fase de ejecución => múltiples unidades de ejecución o unidades de ejecución segmentadas (pipelined execution units).
- Scoreboard verifica y sigue las dependencias, estado u operaciones.
- Scoreboard reemplaza ID, EX, WB por 4 etapas IS (Issue), RO (read operand), EX (Execute) y WB (Write Back).

SCOREBOARD: IMPLICACIONES

1. **Emisión (Issue)**—descodificación de instrucciones & verificación de riesgos estructurales (ID1)

   - Si una unidad funcional capaz de ejecutar la instrucción está libre y ninguna otra instrucción activa tiene el mismo registro de destino (no hay riesgo estructural ni WAW), entonces:
     - el scoreboard emite la instrucción a la unidad funcional y
     - actualiza su estructura de datos interna.
   - En caso contrario (existe un riesgo estructural o WAW),
     - la emisión de la instrucción se detiene, y
     - ninguna otra instrucción más será emitida hasta que estos riesgos desaparezcan.

ETAPAS DE CONTROL EN SCOREBOARD

2. **Lectura de operandos (RO)** —espera hasta que no existan riesgos de datos, entonces se leen los operandos (ID2)

   - Un operando fuente esta disponible si ninguna instrucción emitida activa va a escribir en él, o si el registro que contiene el operando está siendo escrito actualmente por una unidad funcional activa.
   - Cuando los operandos fuente están disponibles, el scoreboard le dice a la unidad funcional que proceda a la lectura de los operandos de los registros y comience la ejecución.
   - El scoreboard resuelve los riesgos RAW dinámicamente en este paso, y las instrucciones pueden ser enviadas a la fase de ejecución out of order.

ETAPAS DE CONTROL EN SCOREBOARD

3. **Ejecución**—operación sobre los operandos (EX)
   - La unidad funcional comienza la ejecución cuando recibe los operandos.
   - Cuando el resultado está listo (ready), notifica al scoreboard que ha completado la ejecución.
4. **Escritura del resultado**—finaliza la ejecución (WB)
   - Una vez que el scoreboard advierte que la unidad funcional ha completado su ejecución, el scoreboard comprueba los riesgos WAR.
   - Si no hay riesgos WAR, escribe el resultado.
   - Si hay riesgo WAR, entonces detiene la instrucción.

   Ejemplo:

```
divd   F0,F2,F4
addd   F10,F0,F8
subd   F8,F8,F14
```

   El scoreboard del CDC 6600 debe detener subd hasta que addd lea los operandos

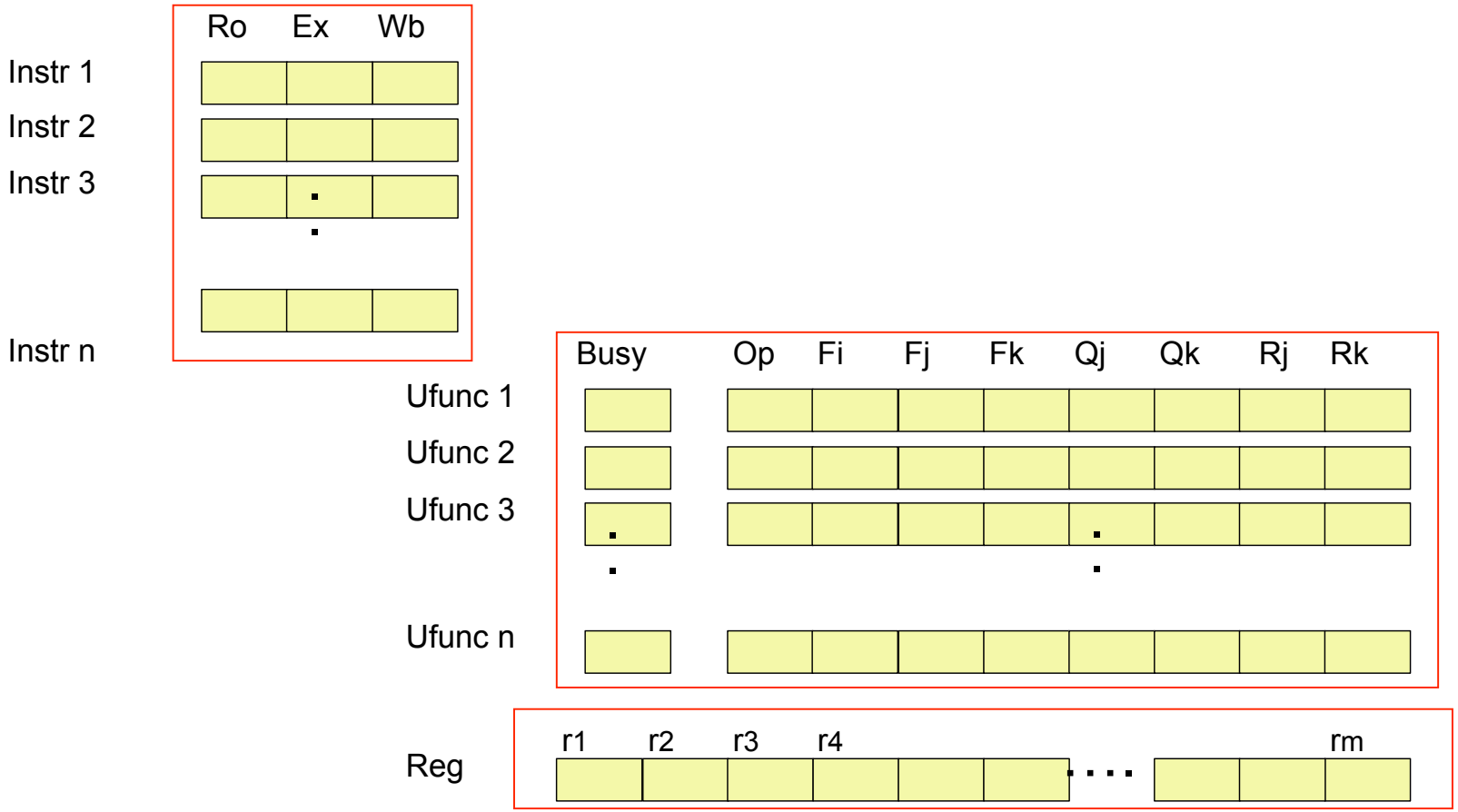
OBSERVACIONES A SCOREBOARD

- La técnica de Scoreboard es del tipo emisión-única, es decir sólo permite emitir una instrucción por ciclo.
- Un riesgo WAW o un riesgo estructural (UF no disponible) provocan que la instrucción en la fase IS se detenga y no pase a RO.
- Un riesgo RAW provoca la detención de las instrucciones en la fase RO y que no pasen a la fase EX.
- Un riesgo WAR provoca que el resultado de una operación no se escriba en el banco de registros en la etapa Wb.
- Después de que una instr se emita hacia una UF, ésta es bloqueada hasta que el resultado de la instrucción en ejecución es escrito en el registro de destino (esto puede relajarse si la escritura en el banco se hace en la primera mitad del ciclo y la lectura en la segunda parte).
- No puede utilizarse la técnica de forwarding o anticipación.
- Los operandos pueden leerse en la etapa RO después de que se han escrito en el banco de registros.

ESTRUCTURA DE DATOS DE SCOREBOARD

- Instruction status—indica en cual de las etapas está la instrucción
- Functional unit status—Indica el estado de la unidad funcional (FU). 9 campos para cada unidad funcional
  - Busy—Indica si la unidad está ocupada o no
  - Op—Operación a realizar en la unidad (e.g., + or –)
  - Fi—Registro de destino
  - Fj, Fk—Número de los registros fuente
  - Qj, Qk—Unidades funcionales que producen los registros fuente Fj, Fk
  - Rj, Rk—Flags indicando cuando están disponibles Fj, Fk
- Register result status—Indica que unidad funcional escribirá en cada registro, si es que existe. En blanco cuando no hay instrucciones pendientes que vayan a escribir en el registro

ESTRUCTURA DE DATOS BÁSICA DEL MARCADOR



ACTUALIZACIÓN DE LOS DATOS EN EL MARCADOR: EJEMPLO

Etapa IS

- **While** inst no emitida **and** inst previa emitida **do**

  if R[dest] = 0 **and** (UF[f] capaz ejecutar Op **and** UF[f,Busy]=0) // f = tipo de unidad funcional

  then

  R[dest] := f;

  UF[f,Busy] := 1;

  Inst[j,RO] := Inst[j,Ex] := Inst[j,Wb] := 0;

  UF[f,Op] := Op; UF[f,Fi] := dest; UF[f,Fj] := src1;

  if R[src1]=0 **then** UF[f,Rj]:=1; // válido

  else UF[f,Rj]:=0;

  UF[f,Qj]:=R[src1]; UF[f,Fk] := src2;

  if R[src2]=0 **then** UF[f,Rk]:=1; // válido

  else UF[f,Rk]:=0;

  UF[f,Qk]:=R[src2];

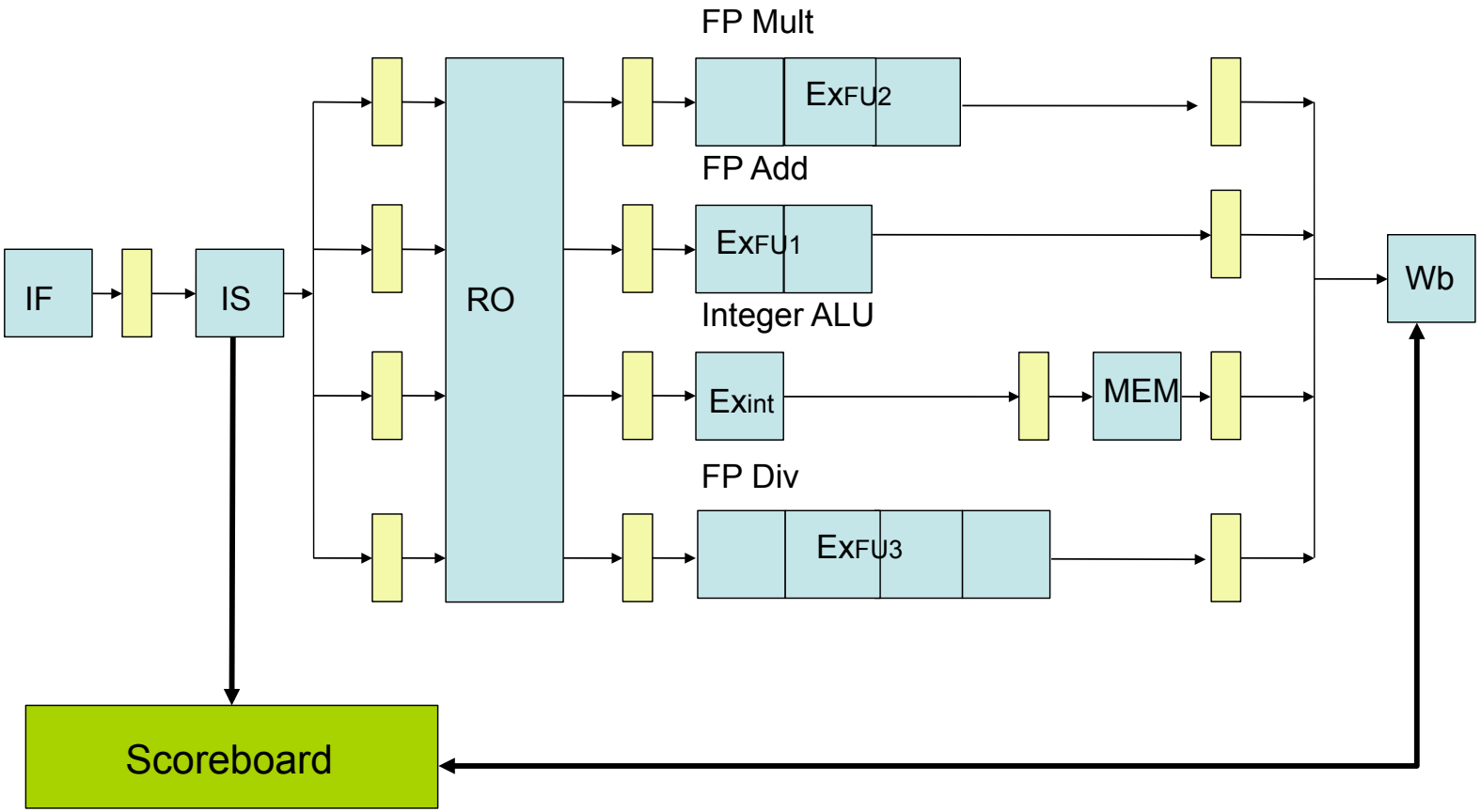
  enddo

CONTROL DEL CAUCE CON SCOREBOARD

Instruction status	Wait until	Bookkeeping
Issue	Not busy (FU) and not result(D)	Busy(FU) ← yes; Op(FU) ← op; Fi(FU) ← `D'; Fj(FU) ← `S1'; Fk(FU) ← `S2'; Qj ← Result( 'S1' ); Qk ← Result(`S2'); Rj ← not Qj; Rk ← not Qk; Result('D') ← FU;
Read operands	Rj and Rk	Rj ← No; Rk ← No
Execution complete	Functional unit done	
Write result	∀f((Fj(f)≠Fi(FU) or Rj(f)=No) & (Fk(f) ≠Fi(FU) or Rk(f)=No))	∀f(if Qj(f)=FU then Rj(f) ← Yes); ∀f(if Qk(f)=FU then Rj(f) ← Yes); Result(Fi(FU)) ← 0; Busy(FU) ← No

*Image from Israel Koren Computer Architecture ECE568 course

CAUCE DLX CON SCOREBOARD

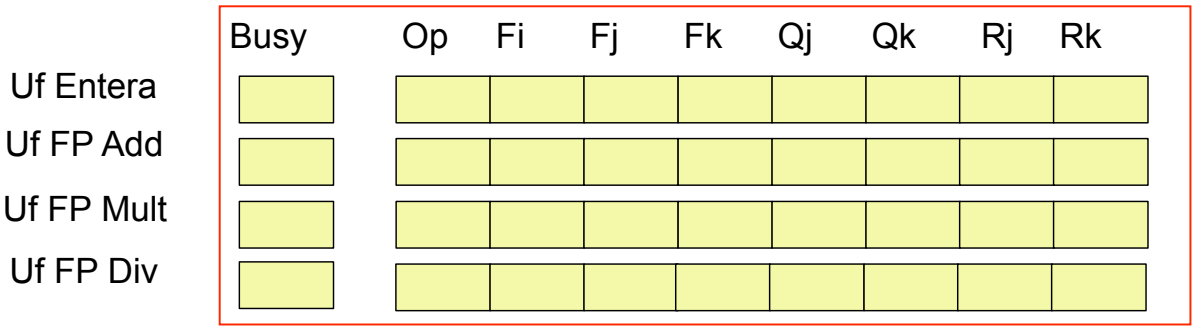


ESTRUCTURA DE DATOS PARA DLX

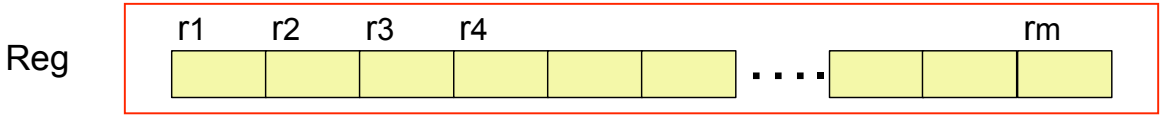
Instr. status

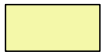
Instr.	If	Is	Ro	Ex	Wb
i1					
i2					
i3					
i4					

Functional Unit status



Register Result status





DLX CON MARCADOR: EJEMPLO

Instr. status

Instr.	If	Is	Ro	Ex	Wb
ld f6,34+,r2					
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	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	No								
Uf FP Add	No								
Uf FP Mult	No								
Uf FP Div	No								

Register Result status

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

Clock

DLX CON MARCADOR: EJEMPLO → CICLO 1

Instr. status

Instr.	If	Is	Ro	Ex	Wb
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

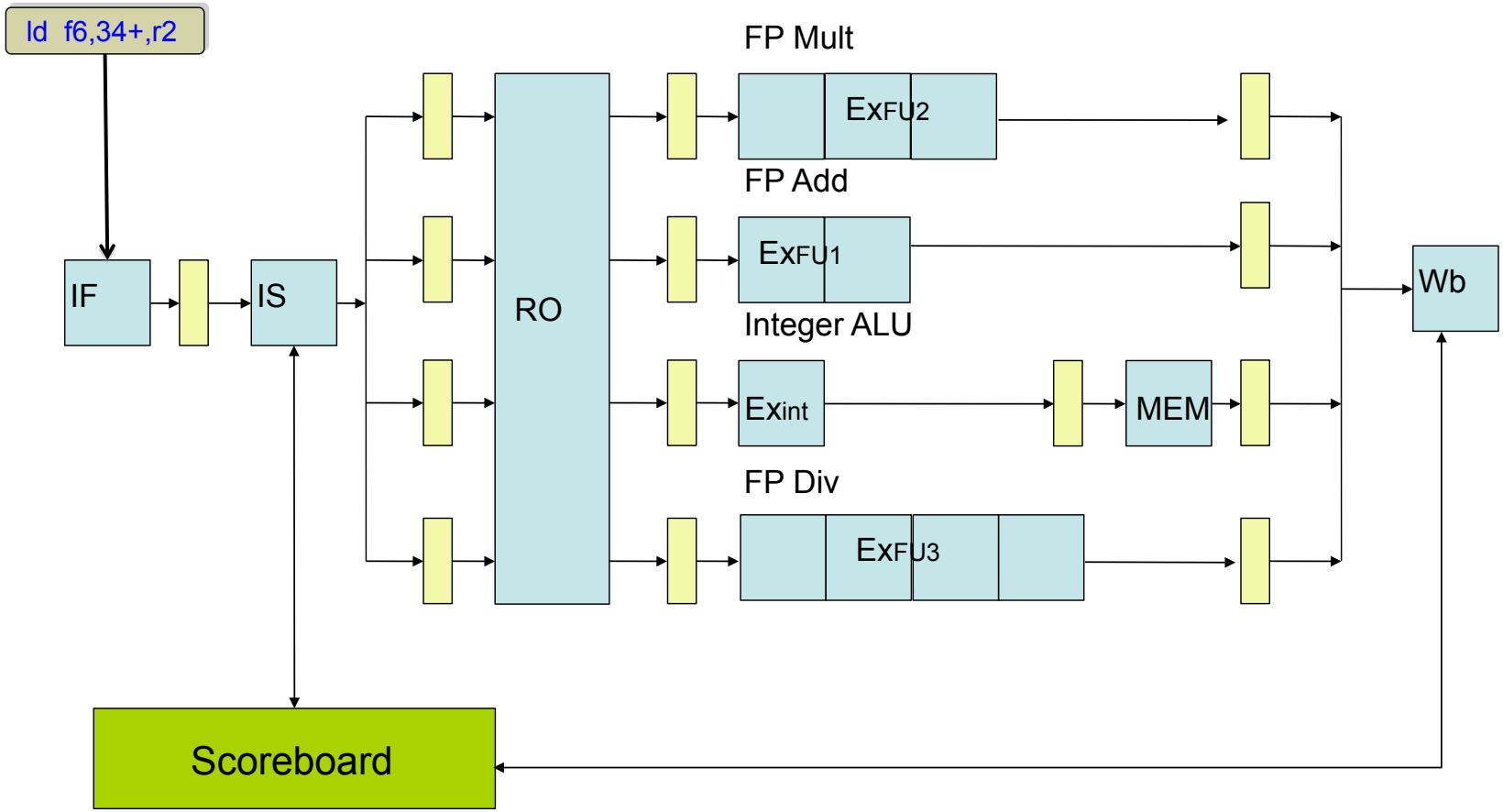
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	No								
Uf FP Add	No								
Uf FP Mult	No								
Uf FP Div	No								

Register Result status

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

Clock 1

DLX CON MARCADOR: EJEMPLO → CICLO 1



DLX CON MARCADOR: EJEMPLO → 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 MARCADOR: EJEMPLO → CICLO 2

Instr. status

Instr.	If	Is	Ro	Ex	Wb
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

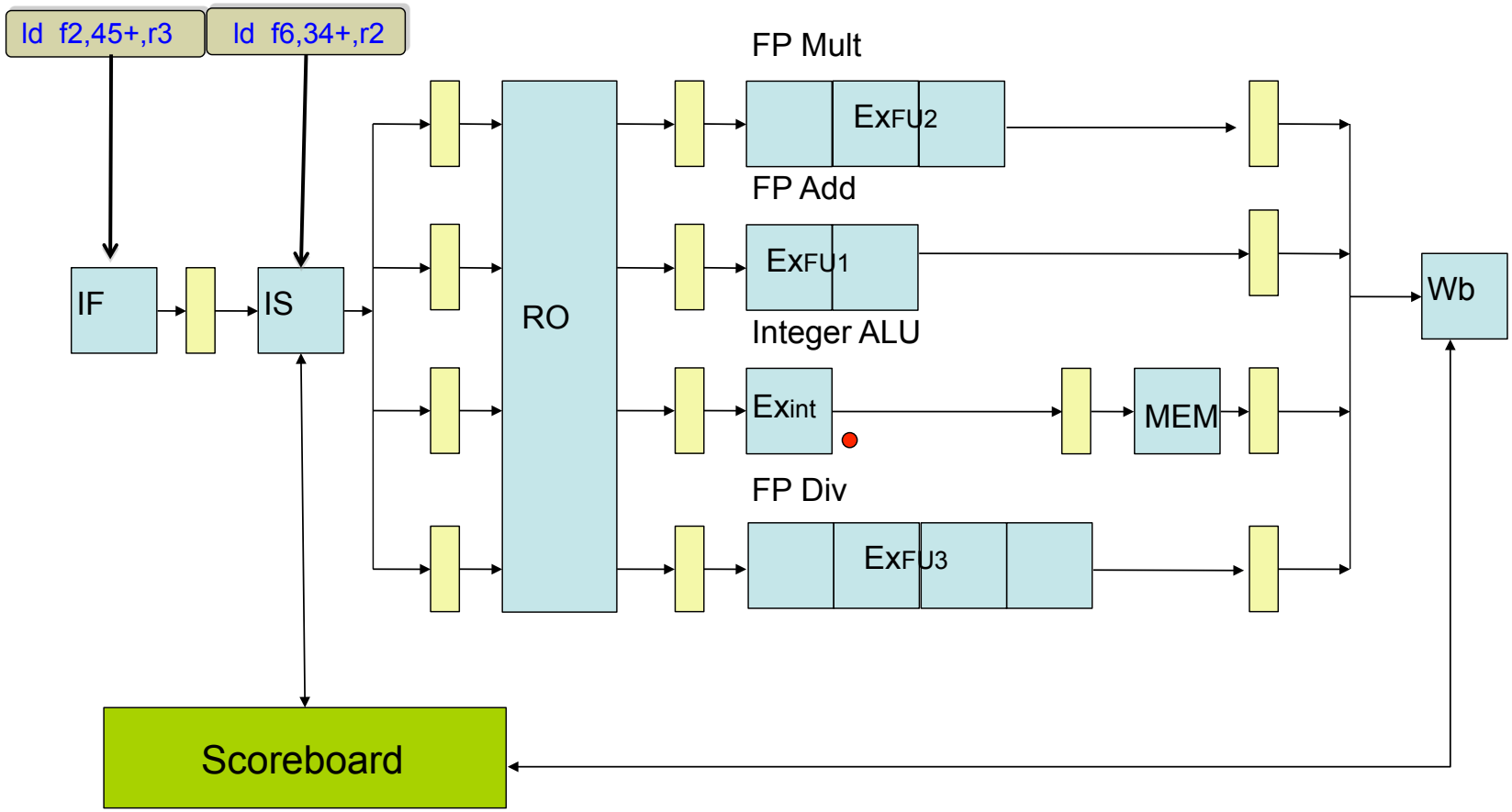
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	Yes	Load	f6		r2				Yes
Uf FP Add	No								
Uf FP Mult	No								
Uf FP Div	No								

Register Result status

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

Clock: 2

DLX CON MARCADOR: EJEMPLO → CICLO 2



- UF reservada

DLX CON MARCADOR: EJEMPLO → 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	Is																		
ld f2,45+,r3		If																		
multd f0,f2,f4																				
subd f8,f6,f2																				
divd f10,f0,f6																				
addd f6,f8,f2																				

DLX CON MARCADOR: EJEMPLO → CICLO 3

Instr. status

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

Functional Unit status

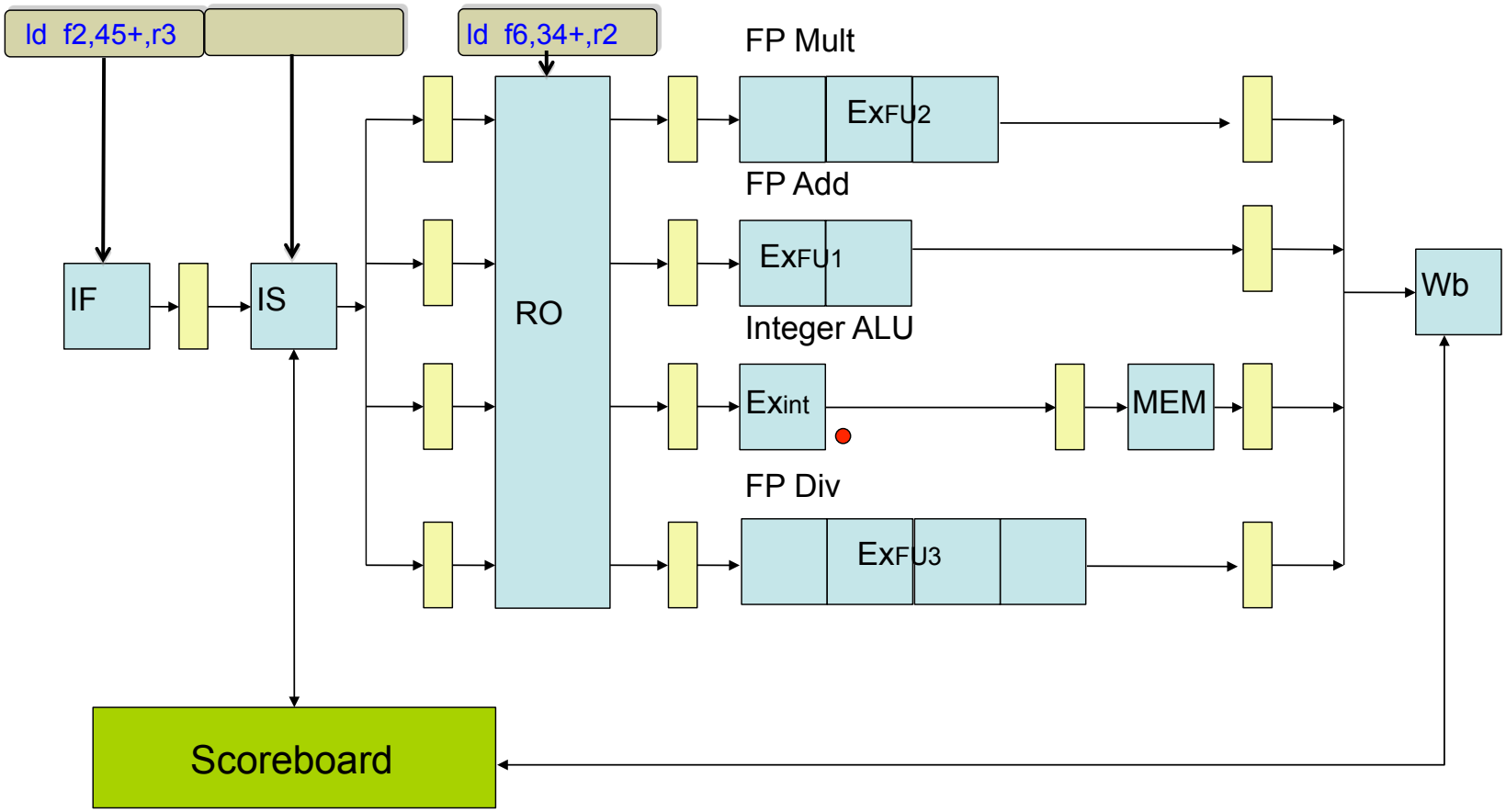
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	Yes	Load	f6		r2				Yes
Uf FP Add	No								
Uf FP Mult	No								
Uf FP Div	No								

Register Result status

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

Clock: 3

DLX CON MARCADOR: EJEMPLO → CICLO 3



DLX CON MARCADOR: EJEMPLO → 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	Is	Ro																	
ld f2,45+,r3		If	o																	
multd f0,f2,f4																				
subd f8,f6,f2																				
divd f10,f0,f6																				
addd f6,f8,f2																				

DLX CON MARCADOR: EJEMPLO → CICLO 4

Instr. status

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

Functional Unit status

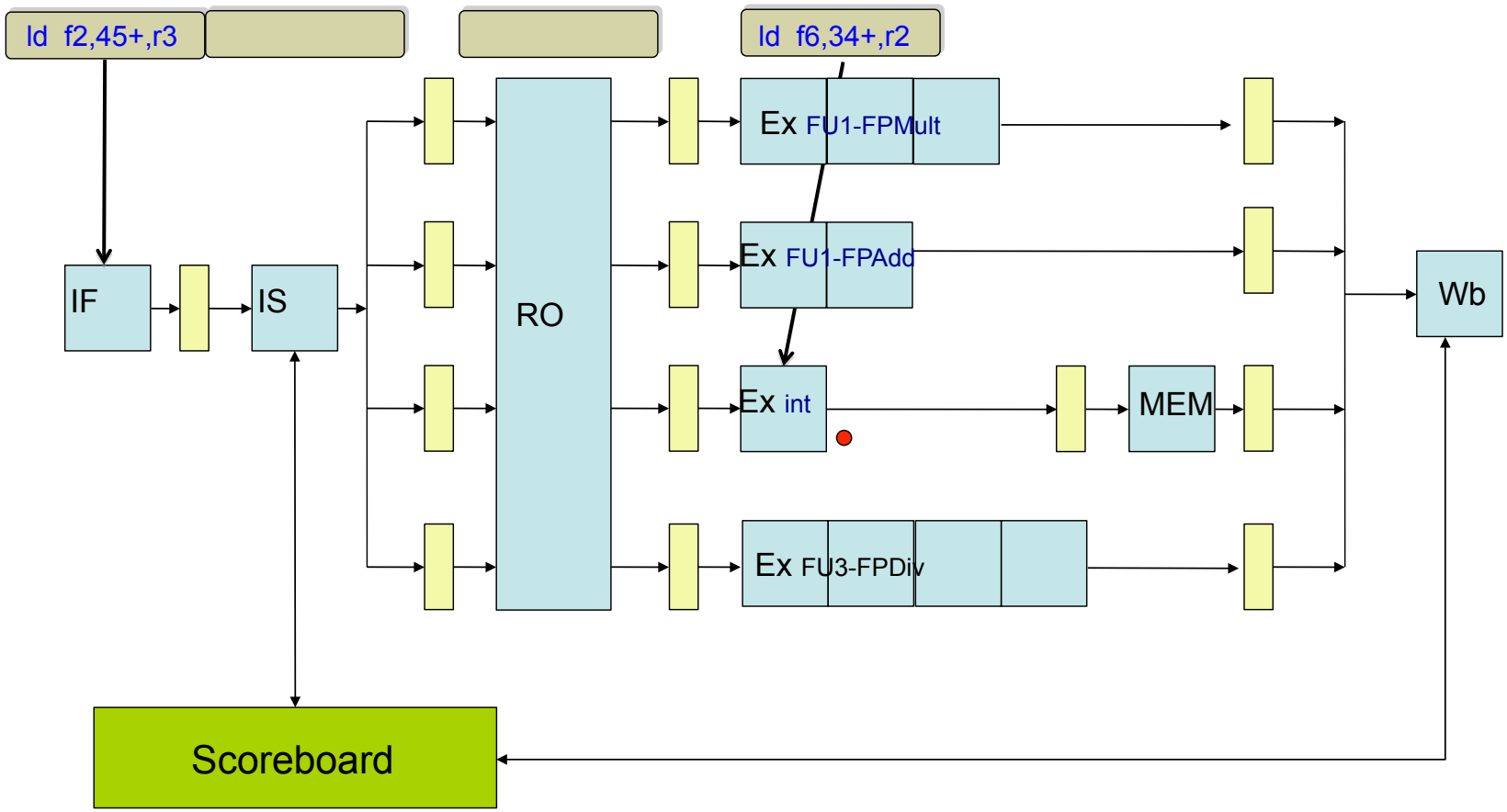
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	Yes	Load	f6		r2				Yes
Uf FP Add	No								
Uf FP Mult	No								
Uf FP Div	No								

Register Result status

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

Clock: 4

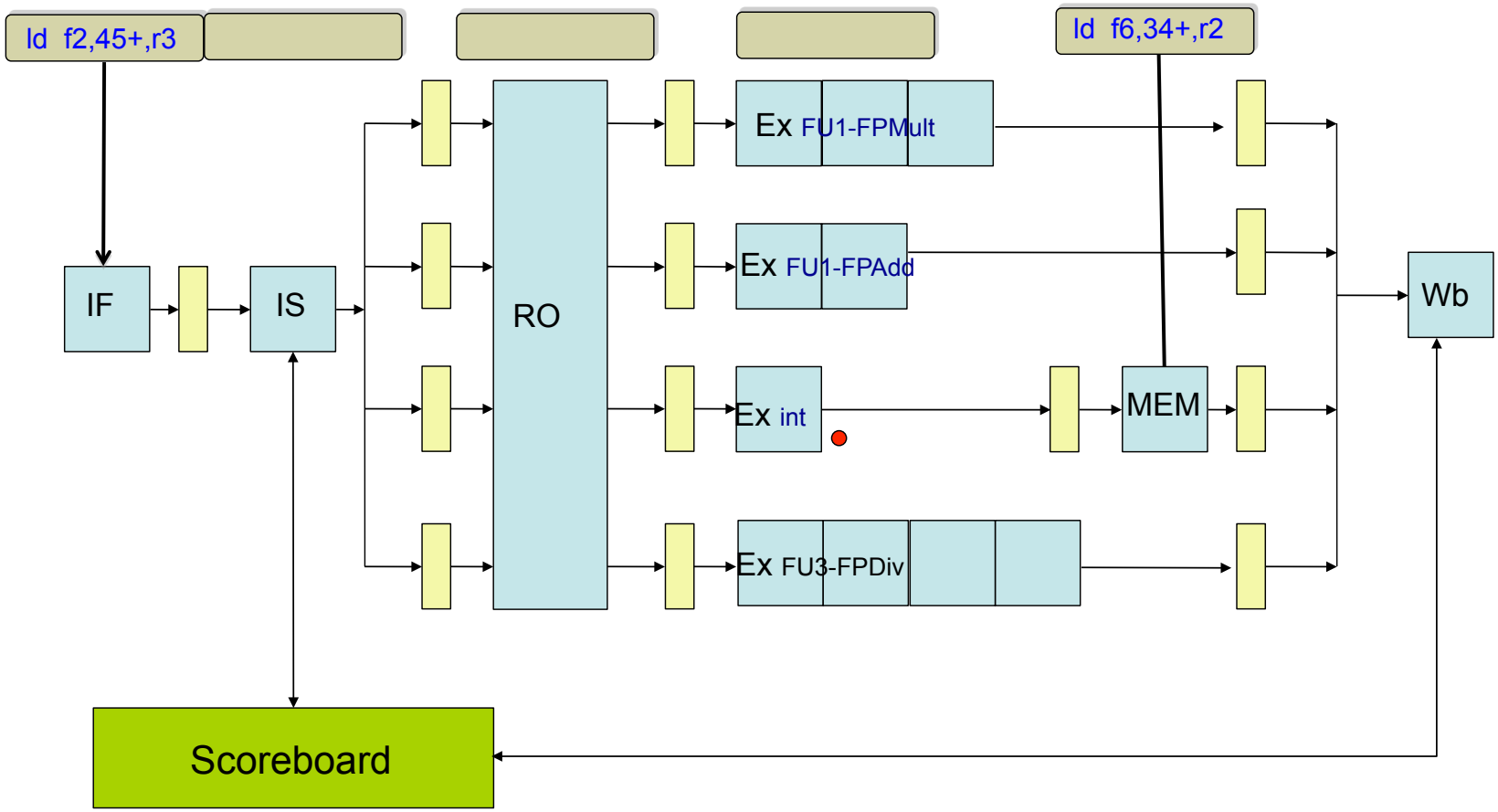
DLX CON MARCADOR: EJEMPLO → CICLO 4



DLX CON MARCADOR: EJEMPLO → 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	Is	Ro	Ex																
ld f2,45+,r3		If	o	o																
multd f0,f2,f4																				
subd f8,f6,f2																				
divd f10,f0,f6																				
addd f6,f8,f2																				

DLX CON MARCADOR: EJEMPLO → CICLO 5



DLX CON MARCADOR: EJEMPLO → 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	Is	Ro	Ex	M															
ld f2,45+,r3		If	o	o	o															
multd f0,f2,f4																				
subd f8,f6,f2																				
divd f10,f0,f6																				
addd f6,f8,f2																				

DLX CON MARCADOR: EJEMPLO → CICLO 6

Instr. status

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

Functional Unit status

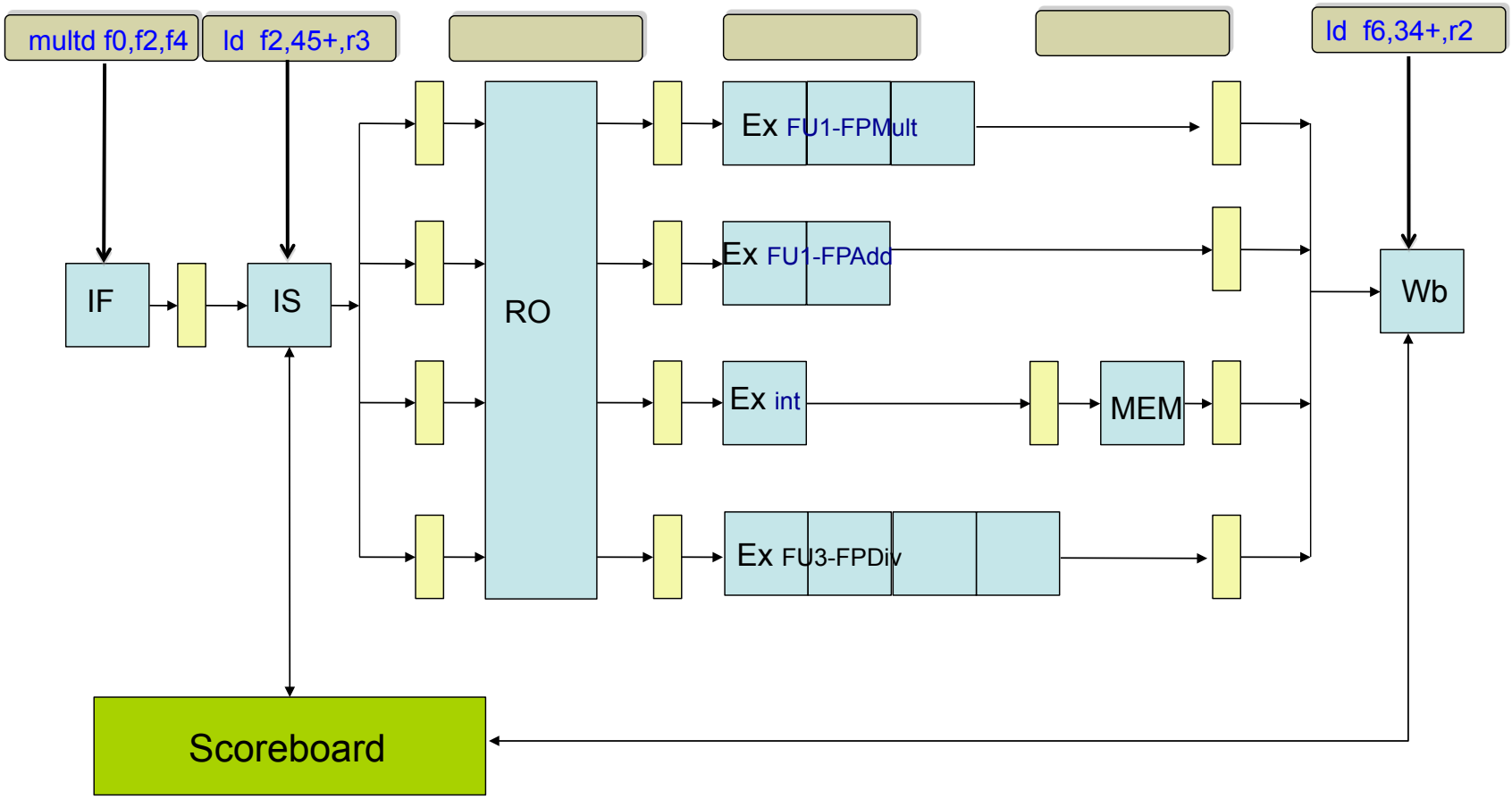
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	Yes	Load	f2		r3				Yes
Uf FP Add	No								
Uf FP Mult	No								
Uf FP Div	No								

Register Result status

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

Clock: 6

DLX CON MARCADOR: EJEMPLO → CICLO 6



DLX CON MARCADOR: EJEMPLO → 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	Is	Ro	Ex	M	W														
ld f2,45+,r3		If	o	o	o	Is														
multd f0,f2,f4						If														
subd f8,f6,f2																				
divd f10,f0,f6																				
addd f6,f8,f2																				

DLX CON MARCADOR: EJEMPLO → CICLO 7

Instr. status

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

Functional Unit status

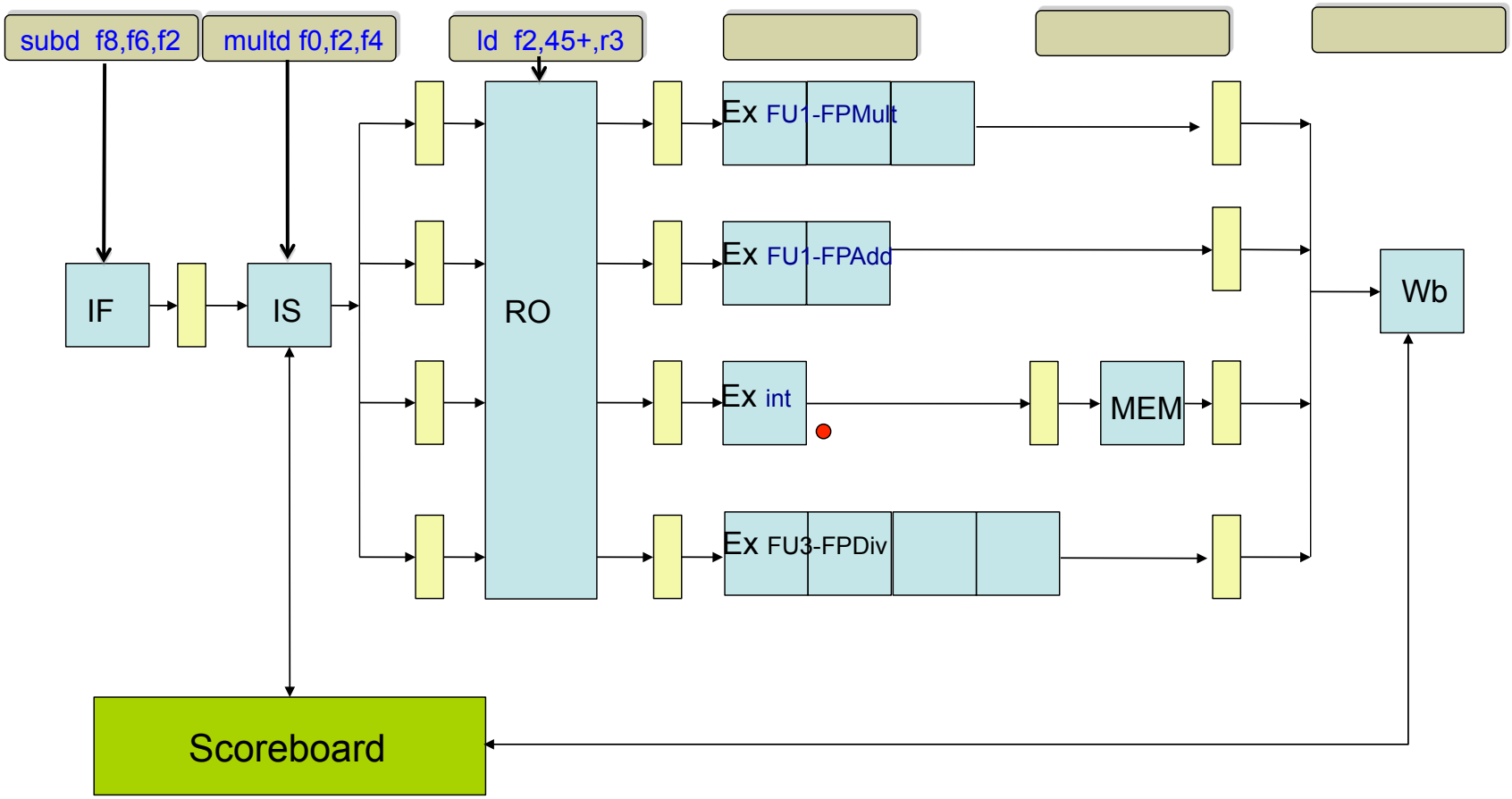
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	Yes	Load	f2		r3				Yes
Uf FP Add	No								
Uf FP Mult	Yes	Mult	f0	f2	f4	UEnt		No	Yes
Uf FP Div	No								

Register Result status

Reg	r0	r1	r2	r3					rm
	f0	f2	f4	f6	f8	f10			fn
	UMul	UEnt							

Clock: 7

DLX CON MARCADOR: EJEMPLO → CICLO 7



DLX CON MARCADOR: EJEMPLO → 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	Is	Ro	Ex	M	W														
ld f2,45+,r3		If	o	o	o	Is	Ro													
multd f0,f2,f4						If	Is													
subd f8,f6,f2							If													
divd f10,f0,f6																				
addd f6,f8,f2																				

DLX CON MARCADOR: EJEMPLO → CICLO 8

Instr. status

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

Functional Unit status

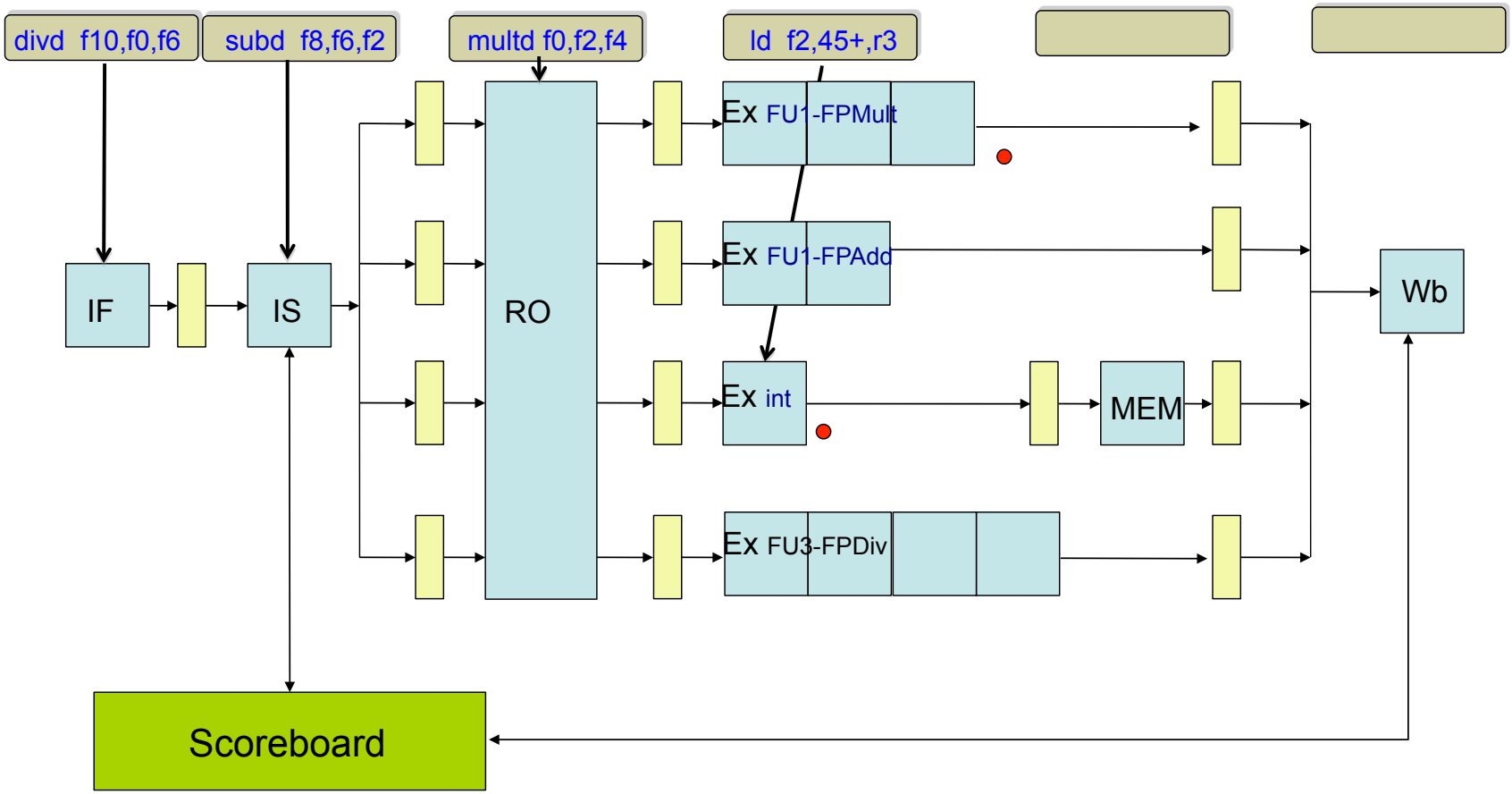
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	Yes	Load	f2		r3				Yes
Uf FP Add	Yes	Sub	f8	f6	f2		UEnt	Yes	No
Uf FP Mult	Yes	Mult	f0	f2	f4	UEnt		No	Yes
Uf FP Div	No								

Register Result status

Reg	r0	r1	r2	r3				rm
	f0	f2	f4	f6	f8	f10		fn
	UMul	UEnt			UAdd			

Clock: 8

DLX CON MARCADOR: EJEMPLO → CICLO 8



DLX CON MARCADOR: EJEMPLO → 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	Is	Ro	Ex	M	W														
ld f2,45+,r3		If	o	o	o	Is	Ro	Ex												
multd f0,f2,f4						If	Is	o												
subd f8,f6,f2							If	Is												
divd f10,f0,f6								If												
addd f6,f8,f2																				

DLX CON MARCADOR: EJEMPLO → CICLO 9

Instr. status

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

Functional Unit status

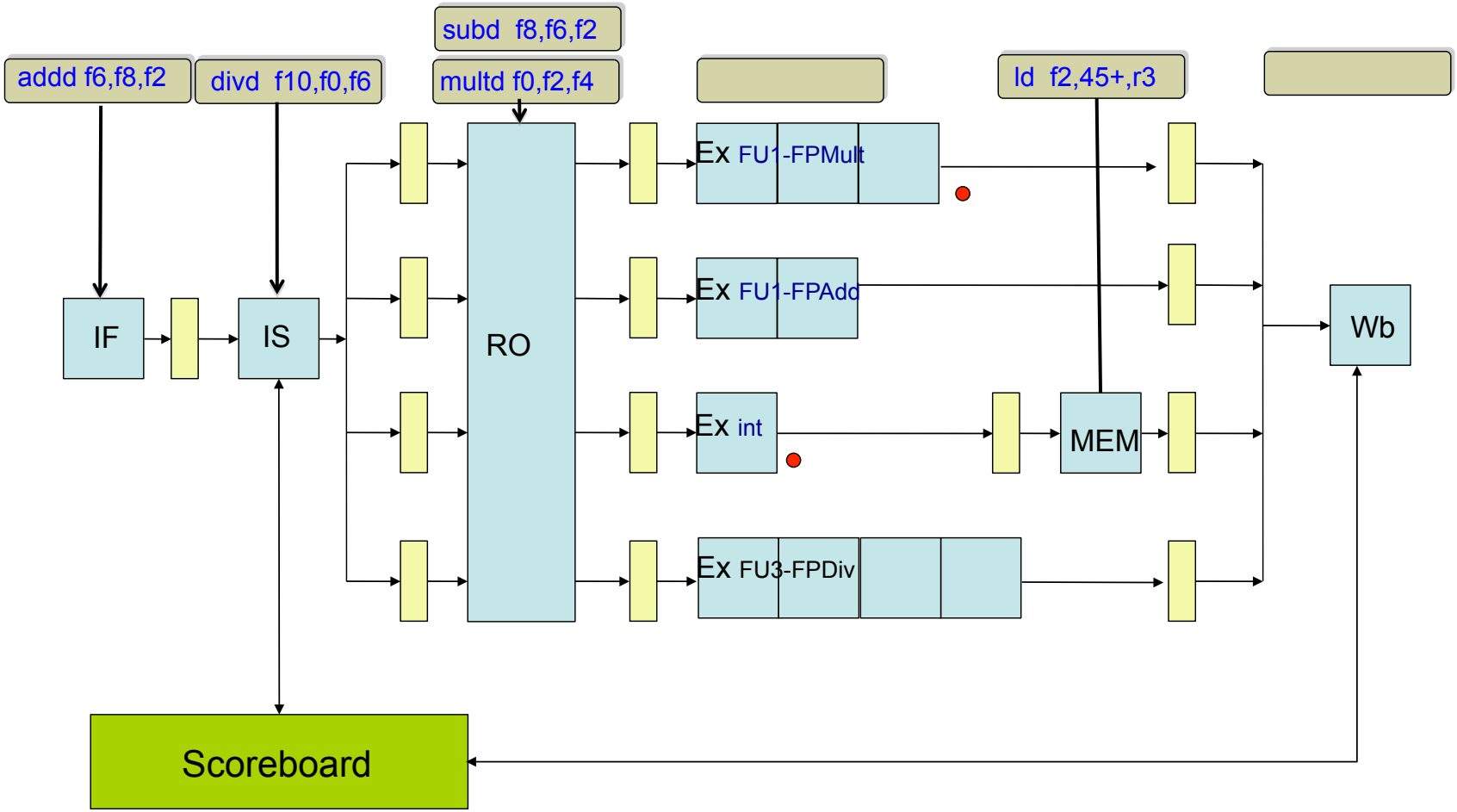
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	Yes	Load	f2		r3				Yes
Uf FP Add	Yes	Sub	f8	f6	f2		UEnt	Yes	No
Uf FP Mult	Yes	Mult	f0	f2	f4	UEnt		No	Yes
Uf FP Div	Yes	Div	f10	f0	f6	UMul		No	Yes

Register Result status

Reg	r0	r1	r2	r3				rm
	f0	f2	f4	f6	f8	f10		fn
	UMul	UEnt			UAdd			

Clock: 9

DLX CON MARCADOR: EJEMPLO → CICLO 9



DLX CON MARCADOR: EJEMPLO → 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	Is	Ro	Ex	M	W														
ld f2,45+,r3		If	o	o	o	Is	Ro	Ex	M											
multd f0,f2,f4						If	Is	o	o											
subd f8,f6,f2							If	Is	o											
divd f10,f0,f6								If	Is											
addd f6,f8,f2									If											

DLX CON MARCADOR: EJEMPLO → CICLO 10

Instr. status

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

Functional Unit status

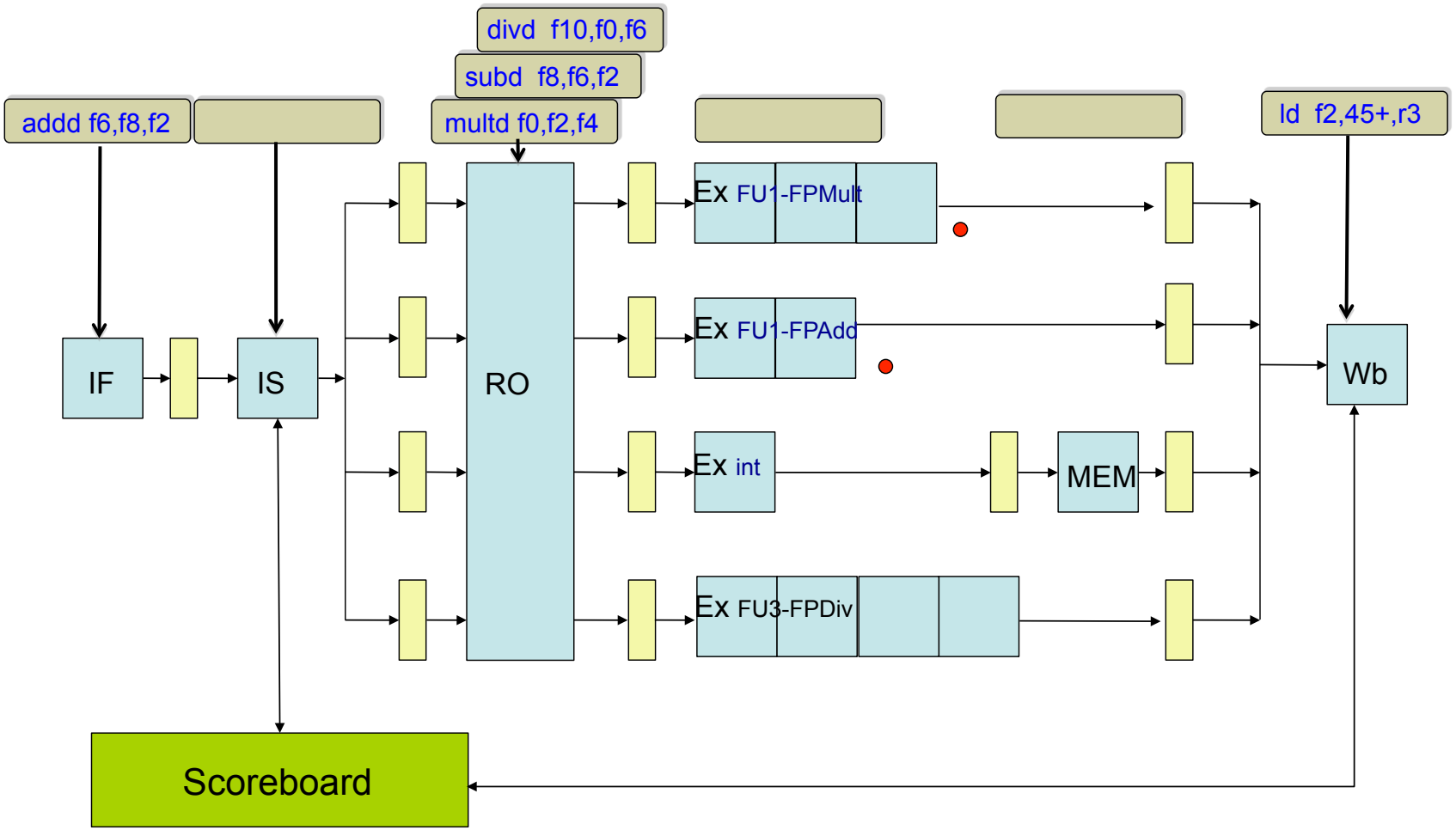
	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	No								
Uf FP Add	Yes	Subd	f8	f6	f2			Yes	Yes
Uf FP Mult	Yes	Mult	f0	f2	f4			Yes	Yes
Uf FP Div	Yes	Div	f10	f0	f6	UMul		No	Yes

Register Result status

Reg	r0	r1	r2	r3	. . . .	rm
	f0	f2	f4	f6	f8	f10	. . . .	fn
	UMul				UAdd	UDiv		

Clock: 10

DLX CON MARCADOR: EJEMPLO → CICLO 10



DLX CON MARCADOR: EJEMPLO → 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	Is	Ro	Ex	M	W														
ld f2,45+,r3		If	o	o	o	Is	Ro	Ex	M	W										
multd f0,f2,f4						If	Is	o	o	Ro										
subd f8,f6,f2							If	Is	o	Ro										
divd f10,f0,f6								If	Is	o										
addd f6,f8,f2									If	o										

DLX CON MARCADOR: EJEMPLO → CICLO 11

Instr. status

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

Functional Unit status

	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	No								
Uf FP Add	Yes	Subd	f8	f6	f2			Yes	Yes
Uf FP Mult	Yes	Mult	f0	f2	f4			Yes	Yes
Uf FP Div	Yes	Divd	f10	f0	f6	UMul		No	Yes

Register Result status

Reg	r0	r1	r2	r3	. . . .	rm
	f0	f2	f4	f6	f8	f10	. . . .	fn
	UMul				UAdd	UDiv		

Clock: 11

DLX CON MARCADOR: EJEMPLO → 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	Is	Ro	Ex	M	W														
ld f2,45+,r3		If	o	o	o	Is	Ro	Ex	M	W										
multd f0,f2,f4						If	Is	o	o	Ro	Ex									
subd f8,f6,f2							If	Is	o	Ro	Ex									
divd f10,f0,f6								If	Is	o	o									
addd f6,f8,f2									If	o	o									

DLX CON MARCADOR: EJEMPLO → CICLO 12

Instr. status

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

Functional Unit status

	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	No								
Uf FP Add	Yes	Subd	f8	f6	f2			Yes	Yes
Uf FP Mult	Yes	Mult	f0	f2	f4			Yes	Yes
Uf FP Div	Yes	Divd	f10	f0	f6	UMul		No	Yes

Register Result status

Reg	r0	r1	r2	r3	...	rm
	f0	f2	f4	f6	f8	f10	...	fn
	UMul				UAdd	UDiv		

Clock: 12

DLX CON MARCADOR: EJEMPLO → CICLO 12

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	Is	Ro	Ex	M	W														
ld f2,45+,r3		If	o	o	o	Is	Ro	Ex	M	W										
multd f0,f2,f4						If	Is	o	o	Ro	Ex	Ex								
subd f8,f6,f2							If	Is	o	Ro	Ex	Ex								
divd f10,f0,f6								If	Is	o	o	o								
addd f6,f8,f2									If	o	o	o								

DLX CON MARCADOR: EJEMPLO → CICLO 13

Instr. status

Instr.	If	Is	Ro	Ex	Wb
ld f6,34+,r2	1	2	3	4	6
ld f2,45+,r3	2	6	7	8	10
multd f0,f2,f4	6	7	10	13	
subd f8,f6,f2	7	8	10	12	13
divd f10,f0,f6	8	9	13		
addd f6,f8,f2	9	13			

Functional Unit status

	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	No								
Uf FP Add	Yes	Addd	f6	f8	f2			Yes	Yes
Uf FP Mult	Yes	Mult	f0	f2	f4			Yes	Yes
Uf FP Div	Yes	Divd	f10	f0	f6	UMul		No	Yes

Register Result status

Reg	r0	r1	r2	r3	...	rm
					...	

	f0	f2	f4	f6	f8	f10	...	fn
	UMul			UAdd		UDiv	...	

Clock: 13

DLX CON MARCADOR: EJEMPLO → 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	Is	Ro	Ex	M	W														
ld f2,45+,r3		If	o	o	o	Is	Ro	Ex	M	W										
multd f0,f2,f4						If	Is	o	o	Ro	Ex	Ex	Ex							
subd f8,f6,f2							If	Is	o	Ro	Ex	Ex	W							
divd f10,f0,f6								If	Is	o	o	o	o							
addd f6,f8,f2									If	o	o	o	Is							

DLX CON MARCADOR: EJEMPLO → CICLO 14

Instr. status

Instr.	If	Is	Ro	Ex	Wb
ld f6,34+,r2	1	2	3	4	6
ld f2,45+,r3	2	6	7	8	10
multd f0,f2,f4	6	7	10	13	14
subd f8,f6,f2	7	8	10	12	13
divd f10,f0,f6	8	9	14		
addd f6,f8,f2	9	13	14		

Functional Unit status

	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	No								
Uf FP Add	Yes	Addd	f6	f8	f2			Yes	Yes
Uf FP Mult	No								
Uf FP Div	Yes	Divd	f10	f0	f6			Yes	Yes

Register Result status

Reg	r0	r1	r2	r3				rm
	f0	f2	f4	f6	f8	f10		fn
				UAdd		UDiv		

Clock: 14

DLX CON MARCADOR: EJEMPLO → 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	Is	Ro	Ex	M	W														
ld f2,45+,r3		If	o	o	o	Is	Ro	Ex	M	W										
multd f0,f2,f4						If	Is	o	o	Ro	Ex	Ex	Ex	W						
subd f8,f6,f2							If	Is	o	Ro	Ex	Ex	W							
divd f10,f0,f6								If	Is	o	o	o	o	Ro						
addd f6,f8,f2									If	o	o	o	Is	Ro						

DLX CON MARCADOR: EJEMPLO → CICLO 15

Instr. status

Instr.	If	Is	Ro	Ex	Wb
ld f6,34+,r2	1	2	3	4	6
ld f2,45+,r3	2	6	7	8	10
multd f0,f2,f4	6	7	10	13	14
subd f8,f6,f2	7	8	10	12	13
divd f10,f0,f6	8	9	14	15	
addd f6,f8,f2	9	13	14	15	

Functional Unit status

	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	No								
Uf FP Add	Yes	Addd	f6	f8	f2			Yes	Yes
Uf FP Mult	No								
Uf FP Div	Yes	Divd	f10	f0	f6			Yes	Yes

Register Result status

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

Clock: 15

DLX CON MARCADOR: EJEMPLO → CICLO 15

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	Is	Ro	Ex	M	W														
ld f2,45+,r3		If	o	o	o	Is	Ro	Ex	M	W										
multd f0,f2,f4						If	Is	o	o	Ro	Ex	Ex	Ex	W						
subd f8,f6,f2							If	Is	o	Ro	Ex	Ex	W							
divd f10,f0,f6								If	Is	o	o	o	o	Ro	Ex					
addd f6,f8,f2									If	o	o	o	Is	Ro	Ex					

DLX CON MARCADOR: EJEMPLO → CICLO 16

Instr. status

Instr.	If	Is	Ro	Ex	Wb
ld f6,34+,r2	1	2	3	4	6
ld f2,45+,r3	2	6	7	8	10
multd f0,f2,f4	6	7	10	13	14
subd f8,f6,f2	7	8	10	12	13
divd f10,f0,f6	8	9	14	16	
addd f6,f8,f2	9	13	14	16	

Functional Unit status

	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	No								
Uf FP Add	Yes	Addd	f6	f8	f2			Yes	Yes
Uf FP Mult	No								
Uf FP Div	Yes	Divd	f10	f0	f6			Yes	Yes

Register Result status

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

Clock: 16

DLX CON MARCADOR: EJEMPLO → CICLO 16

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	Is	Ro	Ex	M	W														
ld f2,45+,r3		If	o	o	o	Is	Ro	Ex	M	W										
multd f0,f2,f4						If	Is	o	o	Ro	Ex	Ex	Ex	W						
subd f8,f6,f2							If	Is	o	Ro	Ex	Ex	W							
divd f10,f0,f6								If	Is	o	o	o	o	Ro	Ex	Ex				
addd f6,f8,f2									If	o	o	o	Is	Ro	Ex	Ex				

DLX CON MARCADOR: EJEMPLO → CICLO 17

Instr. status

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

Functional Unit status

	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	No								
Uf FP Add	No								
Uf FP Mult	No								
Uf FP Div	Yes	Divd	f10	f0	f6			Yes	Yes

Register Result status

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

Clock: 17

DLX CON MARCADOR: EJEMPLO → CICLO 17

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	Is	Ro	Ex	M	W														
ld f2,45+,r3		If	o	o	o	Is	Ro	Ex	M	W										
multd f0,f2,f4						If	Is	o	o	Ro	Ex	Ex	Ex	W						
subd f8,f6,f2							If	Is	o	Ro	Ex	Ex	W							
divd f10,f0,f6								If	Is	o	o	o	o	Ro	Ex	Ex	Ex			
addd f6,f8,f2									If	o	o	o	Is	Ro	Ex	Ex	W			

DLX CON MARCADOR: EJEMPLO → CICLO 18

Instr. status

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

Functional Unit status

	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	No								
Uf FP Add	No								
Uf FP Mult	No								
Uf FP Div	Yes	Divd	f10	f0	f6			Yes	Yes

Register Result status

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

Clock: 18

DLX CON MARCADOR: EJEMPLO → CICLO 18

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	Is	Ro	Ex	M	W														
ld f2,45+,r3		If	o	o	o	Is	Ro	Ex	M	W										
multd f0,f2,f4						If	Is	o	o	Ro	Ex	Ex	Ex	W						
subd f8,f6,f2							If	Is	o	Ro	Ex	Ex	W							
divd f10,f0,f6								If	Is	o	o	o	o	Ro	Ex	Ex	Ex	Ex		
addd f6,f8,f2									If	o	o	o	Is	Ro	Ex	Ex	W			

DLX CON MARCADOR: EJEMPLO → CICLO 19

Instr. status

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

Functional Unit status

	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Uf Entera	No								
Uf FP Add	No								
Uf FP Mult	No								
Uf FP Div	No								

Register Result status

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

Clock: 19

DLX CON MARCADOR: EJEMPLO → CICLO 19

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	Is	Ro	Ex	M	W														
ld f2,45+,r3		If	o	o	o	Is	Ro	Ex	M	W										
multd f0,f2,f4						If	Is	o	o	Ro	Ex	Ex	Ex	W						
subd f8,f6,f2							If	Is	o	Ro	Ex	Ex	W							
divd f10,f0,f6								If	Is	o	o	o	o	Ro	Ex	Ex	Ex	Ex	W	
addd f6,f8,f2									If	o	o	o	Is	Ro	Ex	Ex	W			

SCOREBOARD: LIMITACIONES

- Limitaciones del scoreboard:
  - No hay hardware para forwarding.
  - Limitado a instrucciones en bloques básicos (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.

SCOREBOARD CON UF SEGMENTADAS

- Para eliminar la limitaciones del scoreboard básico en el manejo de las unidades segmentadas recordemos que:
  - En scoreboard básico, la UF se libera cuando el resultado generado en la UF entra en la etapa WB (en el primer semiciclo de reloj).
  - Esto implica que en una instrucción larga la UF está asignada durante muchos ciclos.
- Solución
  - Liberar la UF cuando el resultado de la primera etapa segmentada de la unidad funcional entra en el segundo segmento o si no está segmentada la UF cuando pase al WB.
  - En la arquitectura DLX propuesta el cauce aritmético entero está segmentado en dos etapas la EX y la MEM, mientras que las demás UF requieren varios ciclos pero no están segmentadas.
  - Veamos el efecto en el ejemplo anterior (sobre el cronograma).

DLX CON MARCADOR PARA UF SEGMENTADAS: EJEMPLO CRONOGRAMA

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	Is	Ro	Ex	M	W														
ld f2,45+,r3		If	o	o	Is	Ro	Ex	M	W											
multd f0,f2,f4					If	Is	o	o	Ro	Ex	Ex	Ex	W							
subd f8,f6,f2						If	Is	o	Ro	EX	Ex	W								
divd f10,f0,f6							If	Is	o	o	o	o	Ro	Ex	Ex	Ex	Ex	W		
addd f6,f8,f2								Is	o	o	o	Is	Ro	Ex	Ex	W				

La ganacia se limita a un ciclo ya que la única UF segmentada es la entera y sólo hay dos instrucciónes load.

SUMARIO

- Paralelismo a nivel de instrucción (ILP) en SW o HW.
- Paralelismo a nivel de bucle es más fácil de ver.
- Dependencias del paralelismo SW definidas por programa, riesgos si no pueden ser resueltas por el HD.
- Las dependencias software y la sofisticación del compilador determinan si el compilador puede desenrollar los bucles.
  - Las dependencias de memoria son más difíciles de determinar.
- HW explotando el ILP.
  - Funciona cuando no es posible determinar las dependencias en run time.
  - El código para una máquina funciona bien en otra.
- Idea clave del Scoreboard: permite proseguir a instrucciones posteriores a una detenida (Decode => Issue instr & read operands).
  - Permite la ejecución out-of-order => finalización out-of-order.
  - Etapa ID verifica los riesgos estructurales.

REFERENCIAS

- Curso: Professor David A. Patterson. Computer Science 1996.

- Curso: Prof. H.Shall y A. Gluska. Univ. Haifa.