



## Tema 5b: El Procesador

## Problemas del Pipeline

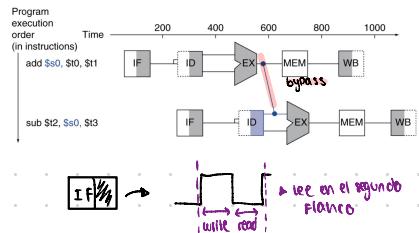
## Peligros en los Datos

una instrucción depende de la anterior

### Soluciones:

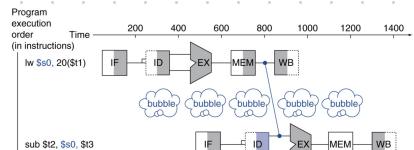
## Forwarding (ByPassing)

antes de guardar el dato → lo pasamos a la siguiente instrucc.



## Caso Load

en el caso de hacer un Iw y necesitar justo después el dato



no puedo tener 2 etapas = en un mismo ciclo de telo)

IF NO!

## Redimension del código

para evitar peligros

## Peligros Estructurales

- conflicto POR  $\rightarrow$  USO RECURSO
  - Pipeline  $\xrightarrow{\text{requiere}}$  2 memorias  $\neq$   datos instrucciones

## Peligros de Control

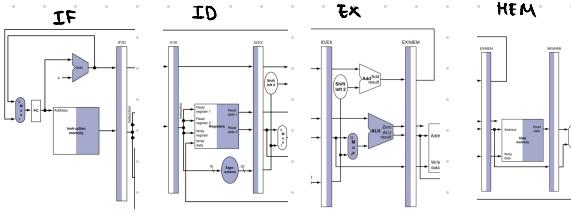
- Branch determinan Flujos del control
    - ↳ coger siguiente instrucción depende Resultado de la RAMC
    - ↳ pipeline no coge la bien la instrucción (o veas sigue en ID) de branch
  - En pipeline de MEPS
    - ↳ Necesita comprobar relojitos + calcular el resultado pronto
    - ↳ añadir hardware para la fase ID

## Operaciones de Pipeline

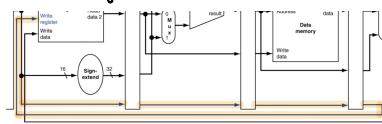
single-clock-cycle  $\Rightarrow$  Muestra Pipeline en 1 Ccl.

Multi-clock-cycle  $\rightarrow$  Muestra Pipeline en varios ciclos

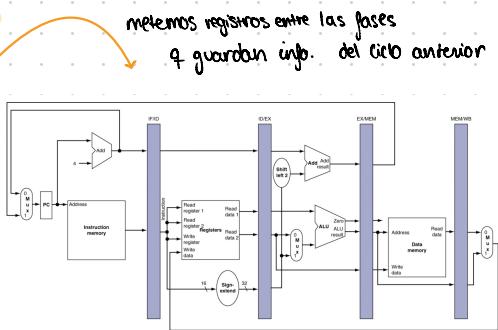
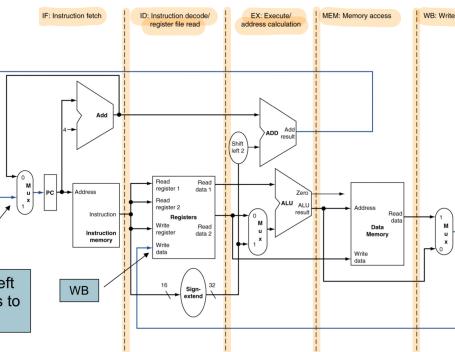
**Load:**



WB (corregido)



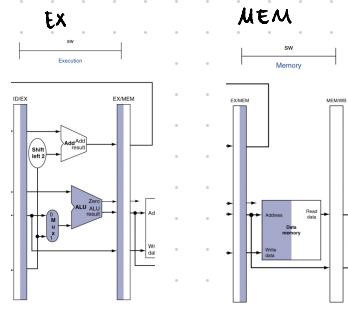
## MIPS PIPELINE DATAPATH



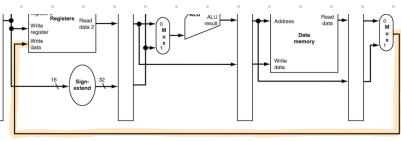
# Tema 5b: El Procesador

## Operaciones de Pipeline

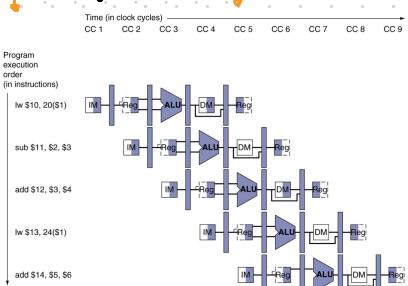
Store:



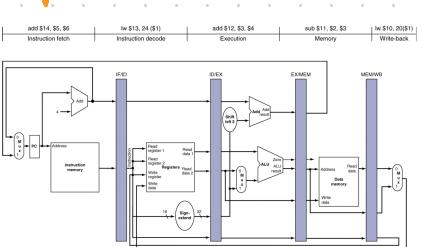
WB



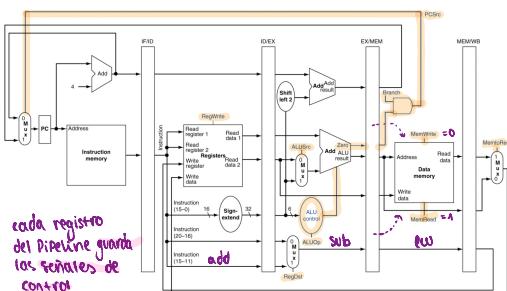
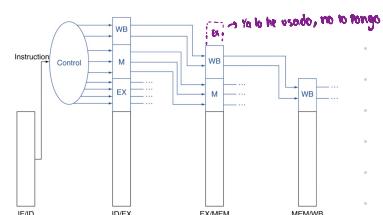
Multi-Cycle Pipeline:



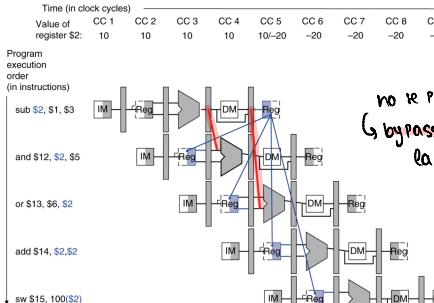
Single-Cycle Pipeline:



control del Pipeline (simplificado):

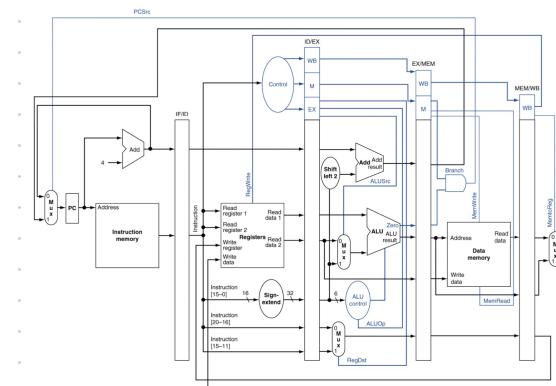
cada registro  
del Pipeline guarda  
los señales de  
control

Dependencias y Forwarding



no se puede ir hacia atrás  
↳ bypass en la fase EX de sub a  
la fase EX de and

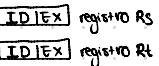
control del Pipeline :



# Tema 5b: El Procesador

## ¿Cuándo necesitamos hacer Forwarding?

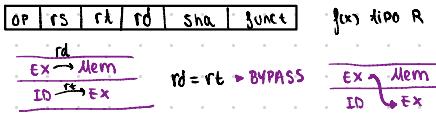
- nº de registro del operando ALU  $\rightarrow$  EX viene dado



### Riesgo de datos cuando

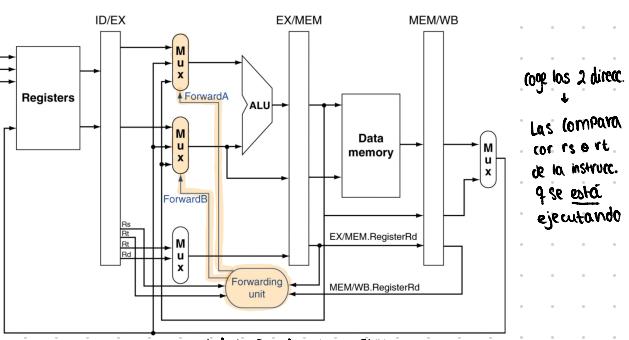
$$\begin{array}{l} \text{EX/MEM reg Rd} = \text{ID/EX reg RS} \quad \{\text{EX} \rightarrow \text{Mem}\} \\ \text{EX/MEM reg Rd} = \text{ID/EX reg RT} \\ \text{ID/EX reg Rd} = \text{ID/EX reg RS} \quad \{\text{Mem} \rightarrow \text{WB}\} \\ \text{ID/EX Reg Rd} = \text{ID/EX reg RT} \end{array}$$

en estos casos  
tengo q volver  
bypass



- SOLO : si la func q recibe BYPASS va a EX en un registro

- SOLO : si Rd ≠ 0



## Condiciones del Forwarding

### Peligro en EX

$$\begin{array}{l} \text{EX/MEM RegWrite} \& \text{EX/MEM RegRd} \neq 0 \\ \& \text{EX/MEM reg Rd} = \text{ID/EX reg RS} \end{array}$$

ForwardA = 10  
Siguiente FASE  
(preguntar)

$$\begin{array}{l} \text{EX/MEM RegWrite} \& \text{EX/MEM RegRd} \neq 0 \\ \& \text{EX/MEM reg Rd} = \text{ID/EX reg RT} \end{array}$$

ForwardB = 10  
Siguiente FASE  
(preguntar)

### Peligro en Mem

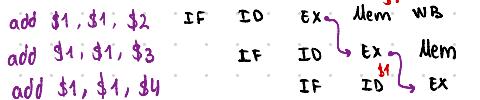
$$\begin{array}{l} \text{Mem WB RegWrite} \& \text{Mem WB RegRd} \neq 0 \\ \& \text{Mem WB reg Rd} = \text{ID/EX reg RS} \end{array}$$

ForwardA = 01  
& NO se cumple

$$\begin{array}{l} \text{Mem WB RegWrite} \& \text{Mem WB RegRd} \neq 0 \\ \& \text{Mem WB reg Rd} = \text{ID/EX reg RT} \end{array}$$

ForwardB = 01  
& NO se cumple

## Peligro de Datos Dobles

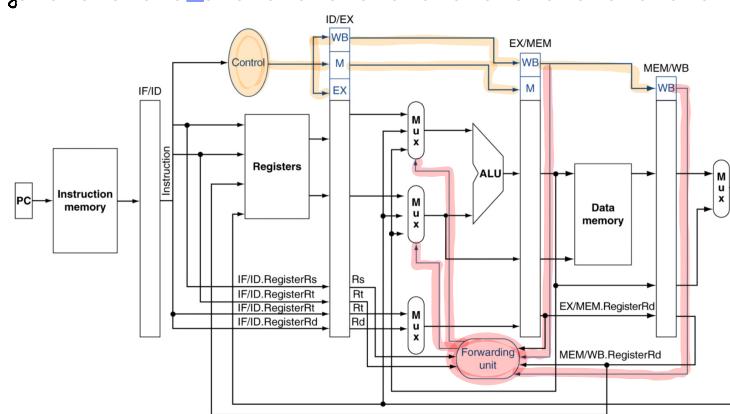


- ambos peligros ocurren  
quieren usar el más reciente

### Revisar la condición de MEM

si solo BYPASS si no hay peligro en EX

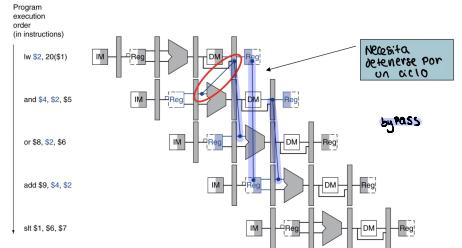
ES el 1er  
peligro que  
hay q mirar



## 4

# Tema 5b: El Procesador

## Peligros Carga-uso [lw-use]



peligros: ? ID cuando la instrucción: decodificada

• nº de registro del operando ALU

↳ ? EX viene dado  
IF | ID reg RS  
IF | ID reg RT

ID | EX | Mem Read

& ID | EX reg RT = IF | ID reg RS → si se detecta: + bubble  
& ID | EX reg RT = IF | ID reg RT  
no By Pass

## ¿Cómo detener el Pipeline?

• Tutar los valores de control en el registro **[ID|EX]** a 0  
↳ EX, Mem j wb: nop (no-operation)

• tutor actualización ↳ CP  
↳ reg **[IF|ID]**

- instruc. se vuelve a decodificar
- se recupera siguiente instruc.
- lo parada de 1-cycle **ponerle** Mem lea datos para lw  
(luego puede pasar a EX)

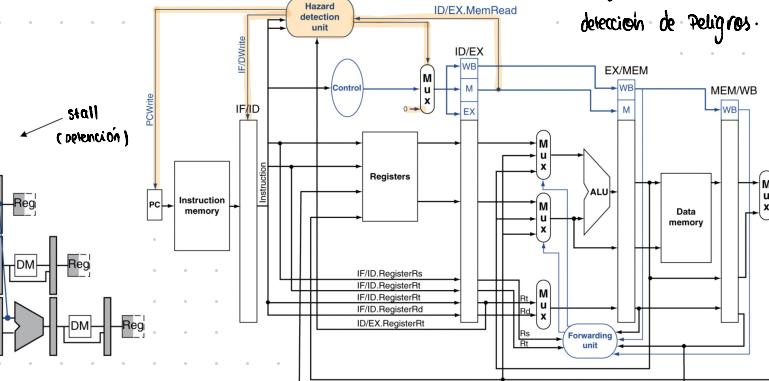
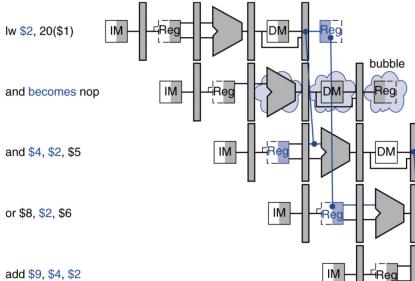
## Ejercicio:

I1 lw \$s1, 0(\$s0)  
I2 lw \$s2, 4(\$s0)  
I3 add \$s3, \$s1, \$s2  
I4 add \$s4, \$s1, \$s3  
I5 lw \$s5, 8(\$s0)  
I6 add \$s6, \$s3, \$s4  
I7 sw \$s6, 12(\$s0)  
I8 beq \$s4, \$s5, label

CON BYPASS

I	1	2	3	4	5	6	7	8	9	10	11	12	13
I1	IF	ID	Ex	Mem	WB								
I2	IF	ID	Ex	Mem	WB								
I3	IF	ID	*	Ex	Mem	WB							
I4		*	*	Ex	Mem	WB							
I5		*	*	IF	*	Ex	Mem	WB					
I6		*	*	IF	ID	*	Ex	Mem	WB				
I7		*	*	IF	*	ID	*	+	Ex	Mem	WB		
I8		*	*	*	*	ID	Ex	Mem	WB				

I	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
I1	IF	ID	Ex	Mem	WB														
I2	IF	ID	Ex	Mem	WB														
I3	IF	ID	*	*	Ex	Mem	WB												
I4		*	*	*	+	*	Ex	Mem	WB										
I5		*	*	*	*	ID	Ex	Mem	WB										
I6		*	*	*	*	IF	ID	*	Ex	Mem	WB								
I7		*	*	*	*	IF	*	ID	*	+	Ex	Mem	WB						
I8		*	*	*	*	*	ID	*	*	*	Ex	Mem	WB						

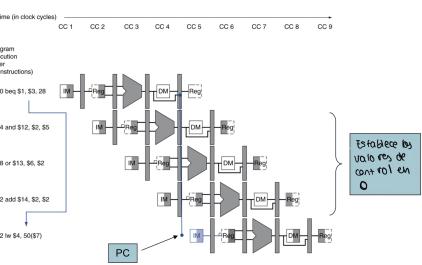


# Tema 5b: El Procesador

## Stalls (detenciones) y Rendimiento

- los stalls → ↓ rendimiento (pero son necesarios)
- el compilador puede organizar el código para evitar → polígonos

- Si el resultado del Branch → está determinado en MUX:

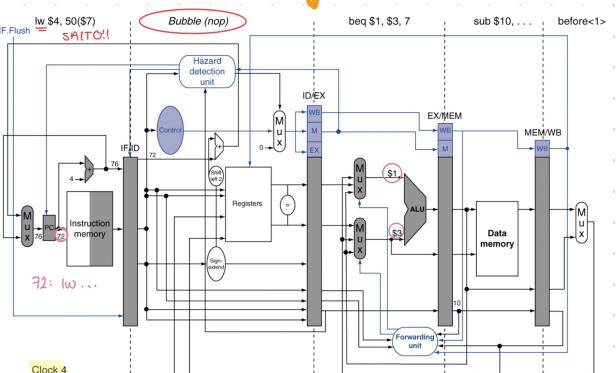
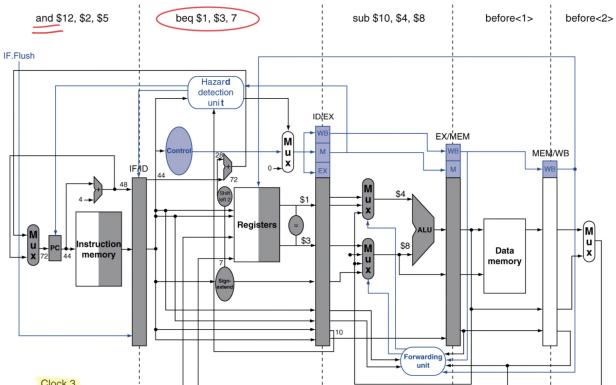


## Reducir el Branch Delay (retardo)

Mover el hardware para del resultado de ID  
↳ sumador de direcc. de destino  
↳ Comparador de registros

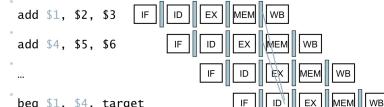
### Ejemplo

	4	2	3
36: sub \$10, \$4, \$8			
40: beq \$1, \$3, 7	IF	ID	EX
44: and \$12, \$2, \$5			
48: or \$13, \$2, \$6			
52: add \$14, \$4, \$2			
56: slt \$15, \$6, \$7			
... 72: lw \$4, 50(\$7)			IF

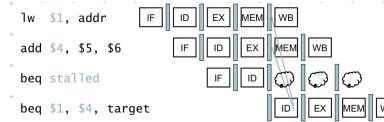


## Peligro de datos Para Branches

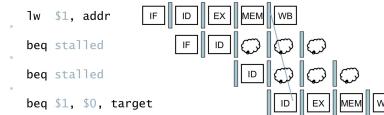
- Si un reg de comparación → es destino C<sub>3,0</sub> instrucción anterior  
Predicción dinámica → si has saltado la últ vez, es prob. q salta ahora



- Si un reg de comparación → es destino C<sub>3,0</sub> instrucción anterior  
↓  
necesita + ciclo de Parada



- Si un reg de comparación → es destino C<sub>3,0</sub> instrucción anterior  
↓  
necesita 2 ciclos de Parada

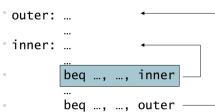


# Tema 5b: El Procesador

## Predicción dinámica del BRANCH

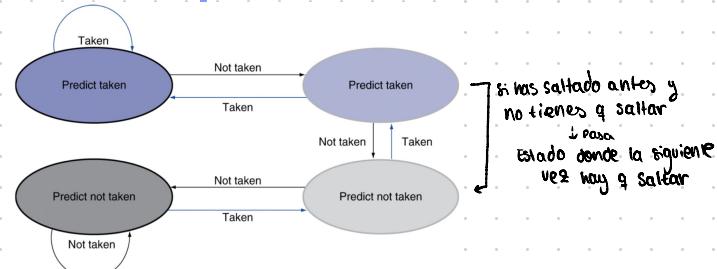
- Predicción con Buffer ("Branch History table")
- Indexado por direcciones de instrucciones. Branch recientes
- Almacena el resultado (taken) / not taken
- Para ejecutar una rama:
  - Revisa la tabla ("table")  $\rightarrow$  espera el = resultado
  - Comienza a  $\downarrow$  desde  $\xrightarrow{\text{Jal-TakeOff}}$  target (objetivo)
  - Si está mal  $\xrightarrow{\text{adios pipeline}}$  cambiar predicción

## Vaticinador ("Predictor") 1 Bit : Shortcoming [defecto]



- Predice erróneamente  $\rightarrow$  "taken" en la últ. iteración del ciclo
- Predice erróneamente  $\rightarrow$  "Not taken" en la 1<sup>a</sup> iteración del ciclo intern O la prox. vez

1 bit no es suficiente para prever 2 bits

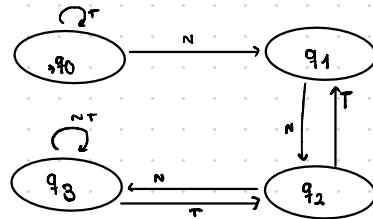


Estado de "No saltes más"  $\leftarrow \dashrightarrow$  'pasa'

## Ejercicio:

Una instrucción de salto tiene la siguiente secuencia de "tomado" (T) y "no tomado" (N): TTTNNNNNTNNNN

¿Es cierto que, por la secuencia anterior de resultados del salto, el predictor tiene una tasa de acierto de 70%?



Estado	q0	q0	q0	q0	q1	q2	q3	q2	q1	q0	q1	q2
Predicción	T	T	T	T	T	N	N	N	T	T	T	N
Resultado	T	T	T	N	N	N	T	T	T	N	N	N