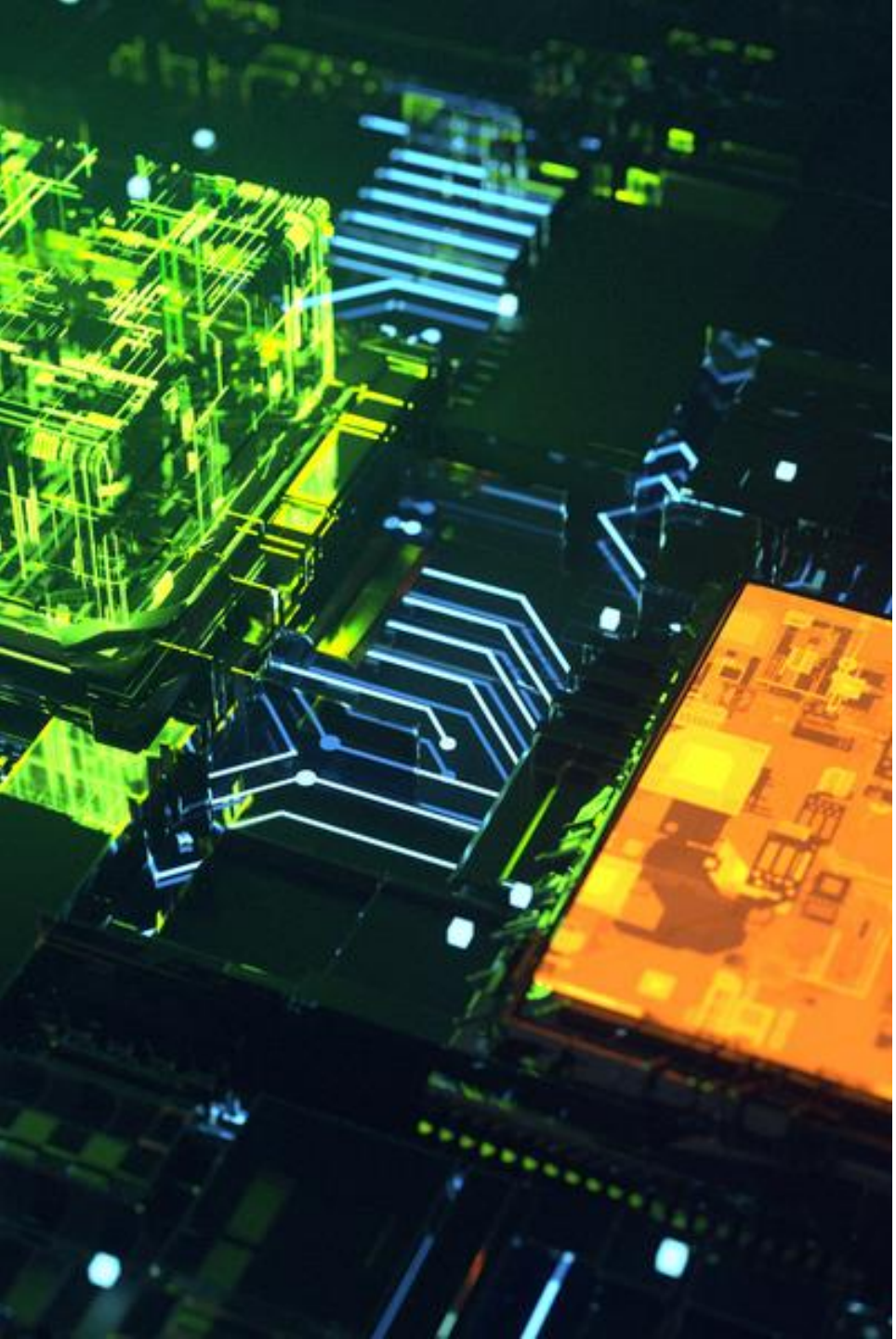


Tema 4

SEGMENTACIÓN

Índice

1. Introducción
2. Segmentación de la ruta de datos
3. Riesgos de segmentación
4. Paralelismo a nivel de instrucciones
5. Conclusiones

Concepto de segmentación

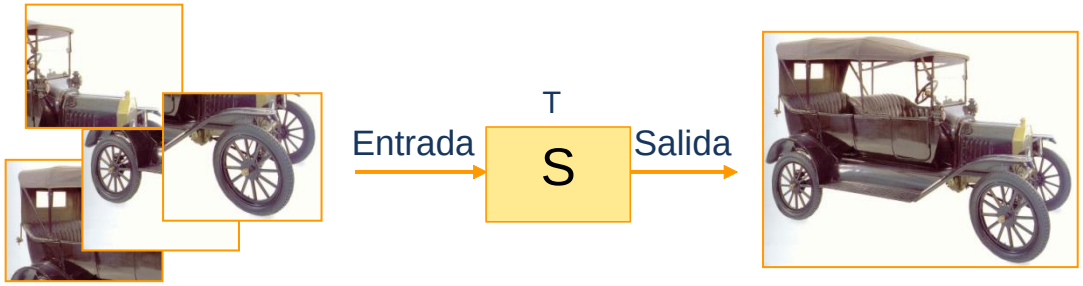
- La segmentación es una de las claves que permite **aumentar** el **rendimiento** en los computadores

- Analogía con una cadena de montaje industrial
  - La ejecución de una tarea se divide en etapas, cada elemento de la cadena se especializa en realizar una operación concreta

- Explota el **paralelismo temporal**
  - Opera de forma serie para una piza determinada
  - Ejecución de varias tareas simultáneamente

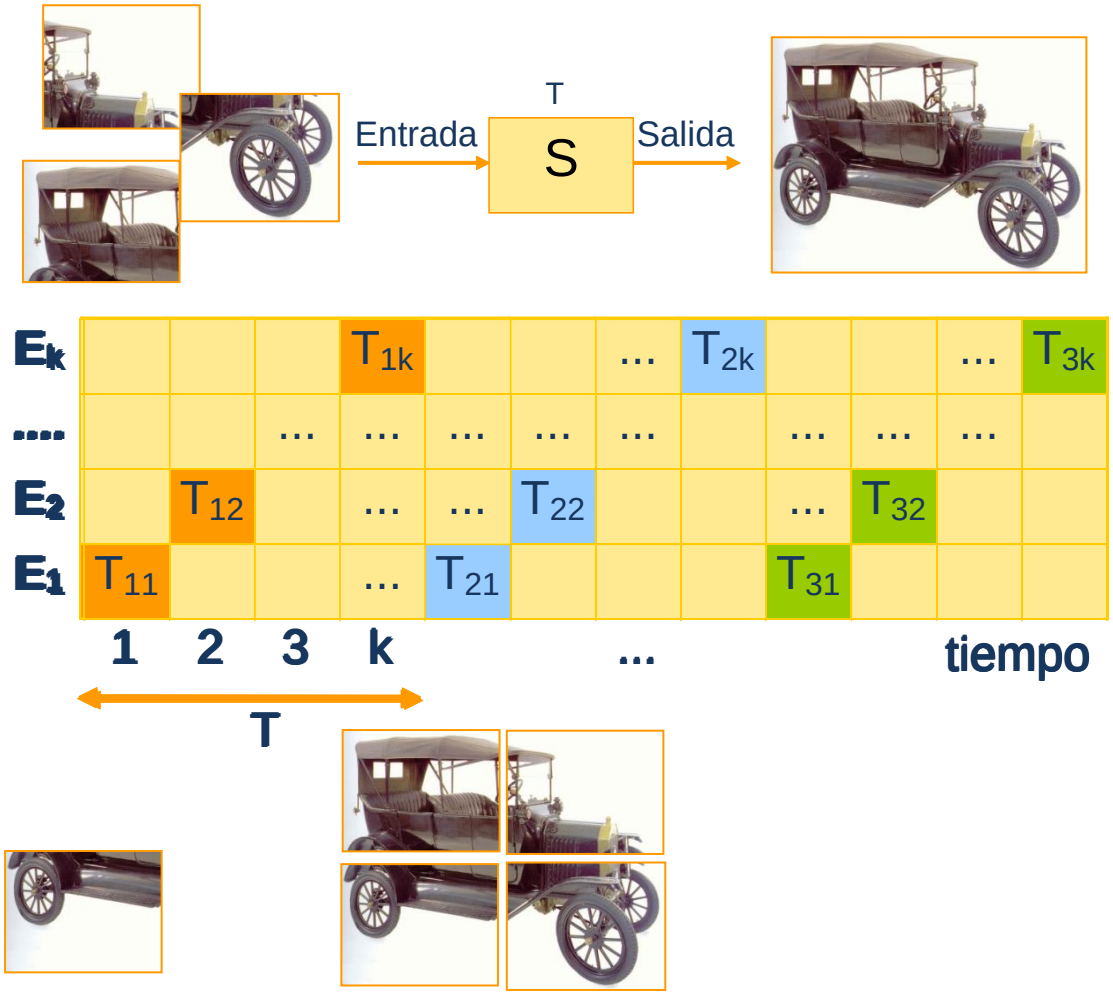
Analogía. Caso secuencial

- Una tarea puede realizarse cuando todos los **recursos** están **disponibles**



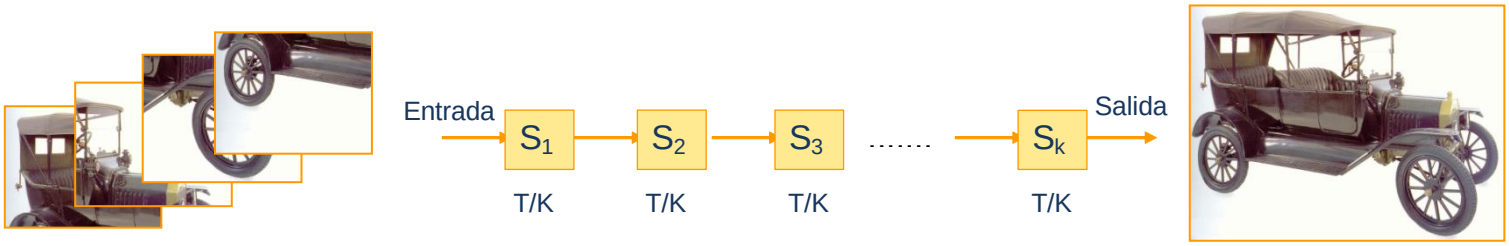
Analogía. Caso secuencial

- Una tarea puede realizarse cuando todos los **recursos** están **disponibles**



Analogía. Caso segmentado

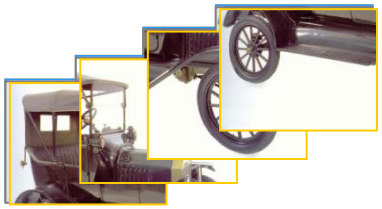
- El comienzo de una tarea en una etapa sólo requiere la finalización de la tarea anterior en esa etapa

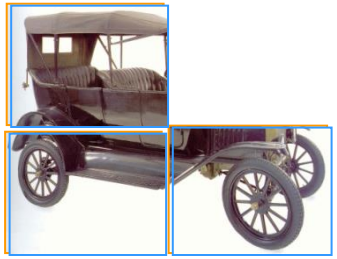


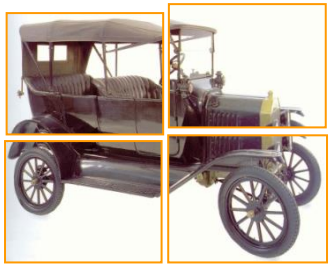
- Cada trabajador pasa al siguiente trabajador de la cadena el estado actual del trabajo mientras puede comenzar a trabajar en el siguiente vehículo

Analogía. Caso segmentado

- El comienzo de una tarea en una etapa sólo requiere la finalización de la tarea anterior en esa etapa

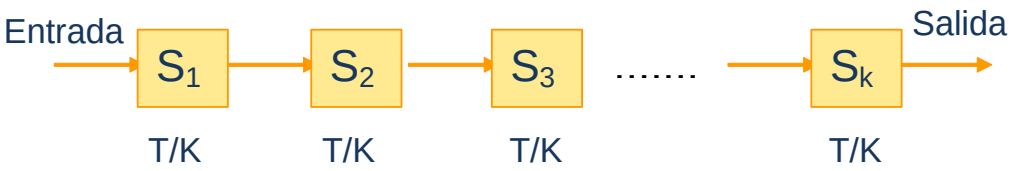






Proceso de segmentación

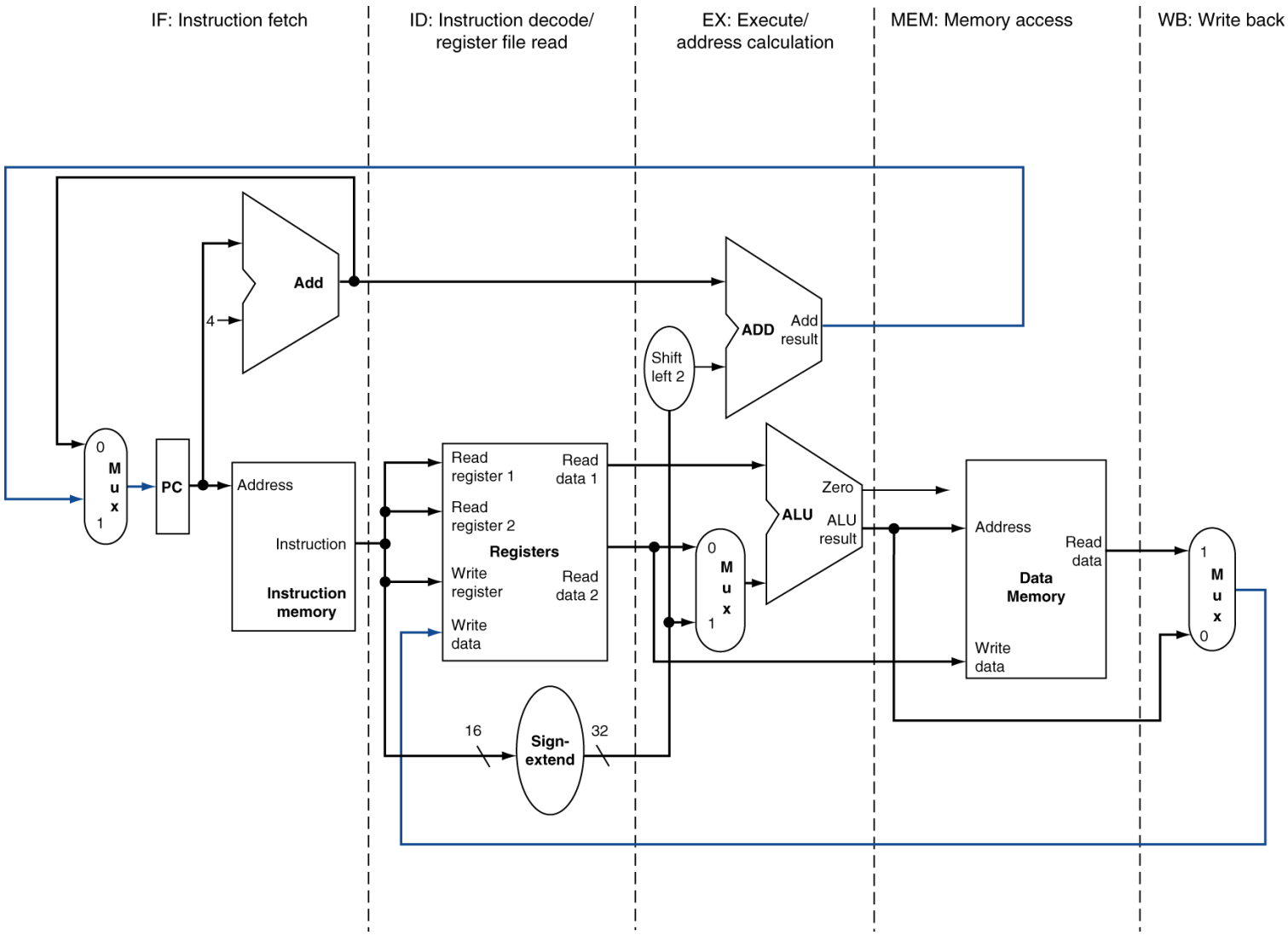
- Factor determinante: descomposición de la tarea a realizar en etapas
  - Distribución uniforme del tiempo (caso ideal)
  - Etapa más lenta actúa como cuello de botella
  - Ajustar el ritmo de trabajo a la etapa más lenta
- Es necesario contemplar:
  - Para que cada trabajador pueda pasar el trabajo necesita tiempo para preparar y distribuir la parte del vehículo que lleva fabricada
  - También tener en cuenta que la nueva organización necesaria para controlar el proceso puede ser muy compleja



2. Segmentación de la ruta de datos

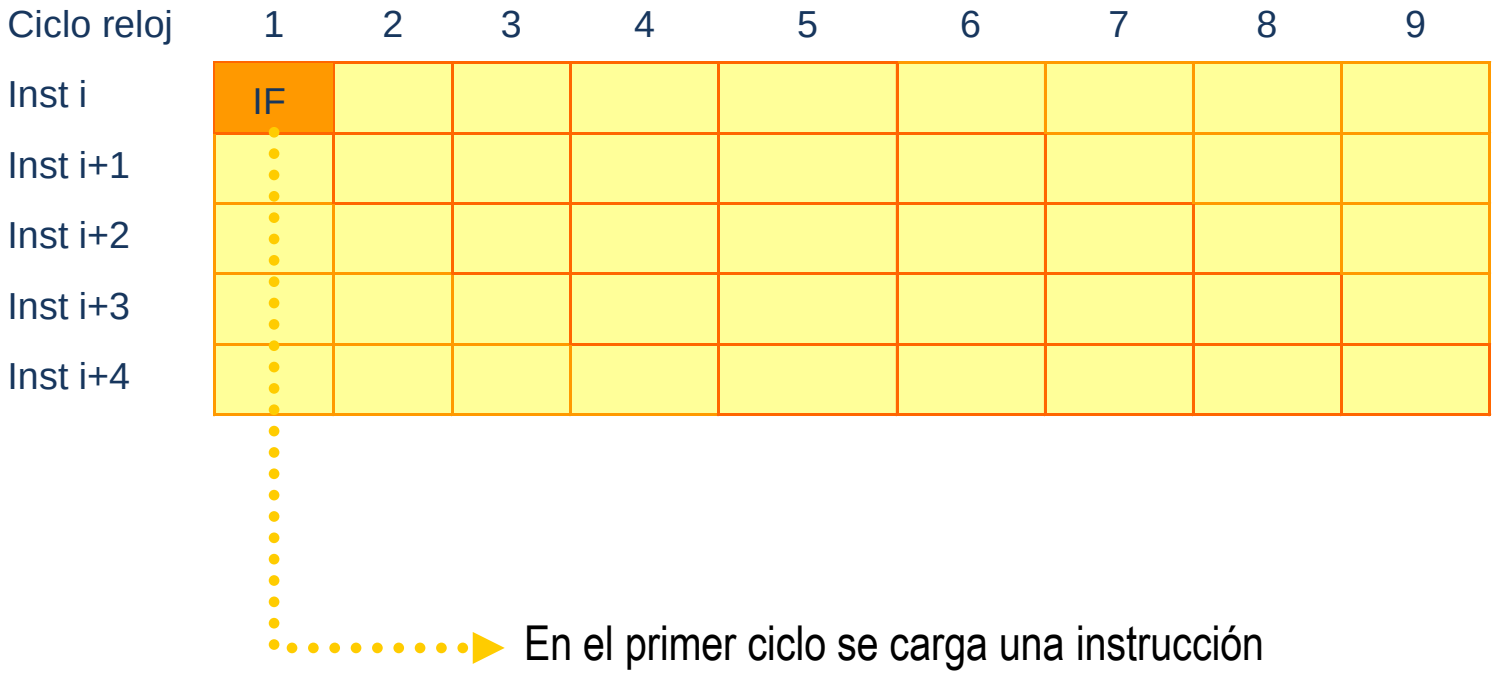
- Definición de las etapas (5 etapas: un paso en cada etapa)
  1. **IF**: Búsqueda de la instrucción (e incremento del PC)
  2. **ID**: Decodificación de la instrucción y lectura de registros
  3. **EX**: Ejecutar operación o calcular direcciones efectivas
  4. **MEM**: Acceso al operando en memoria
  5. **WB**: Escritura del resultado en registro

Distribución de las etapas



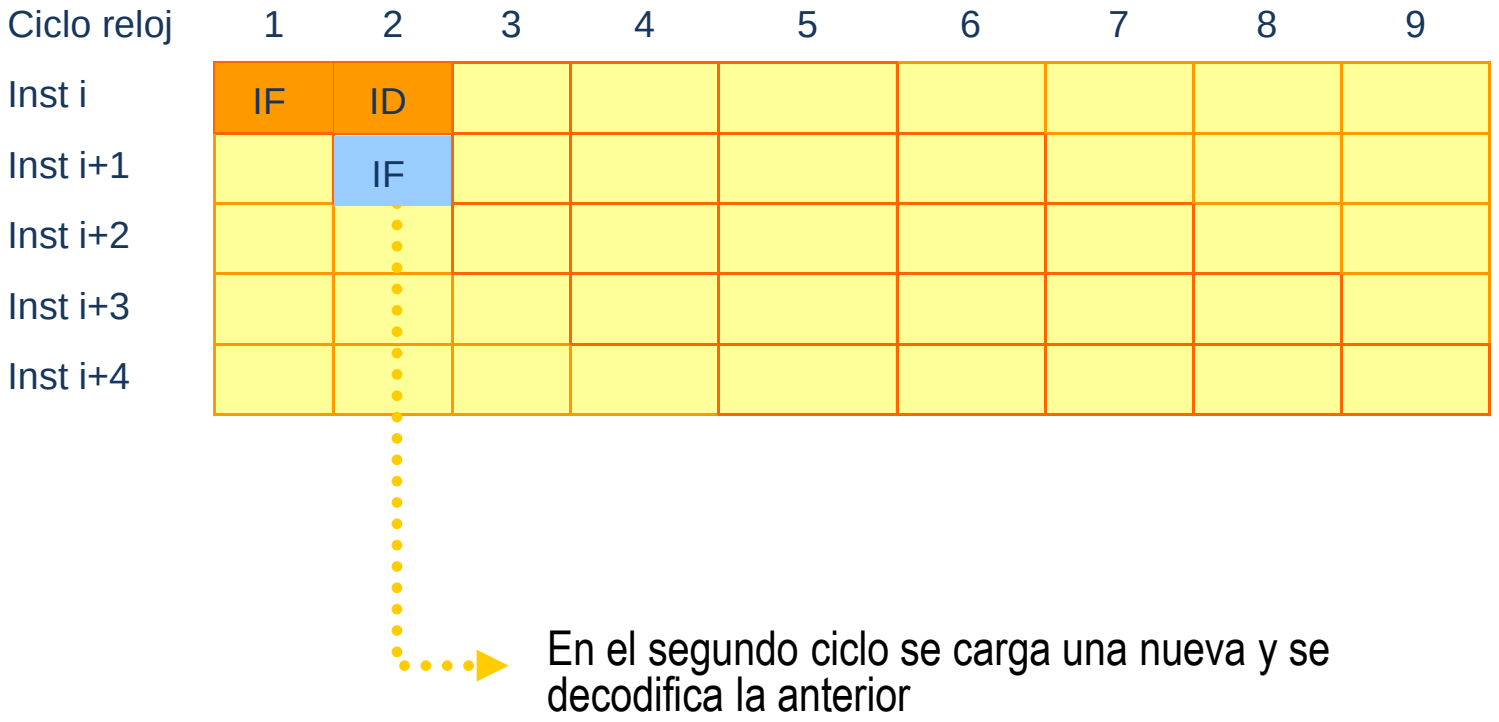
Segmentación de instrucciones

- La segmentación consiste en solapar la ejecución de instrucciones



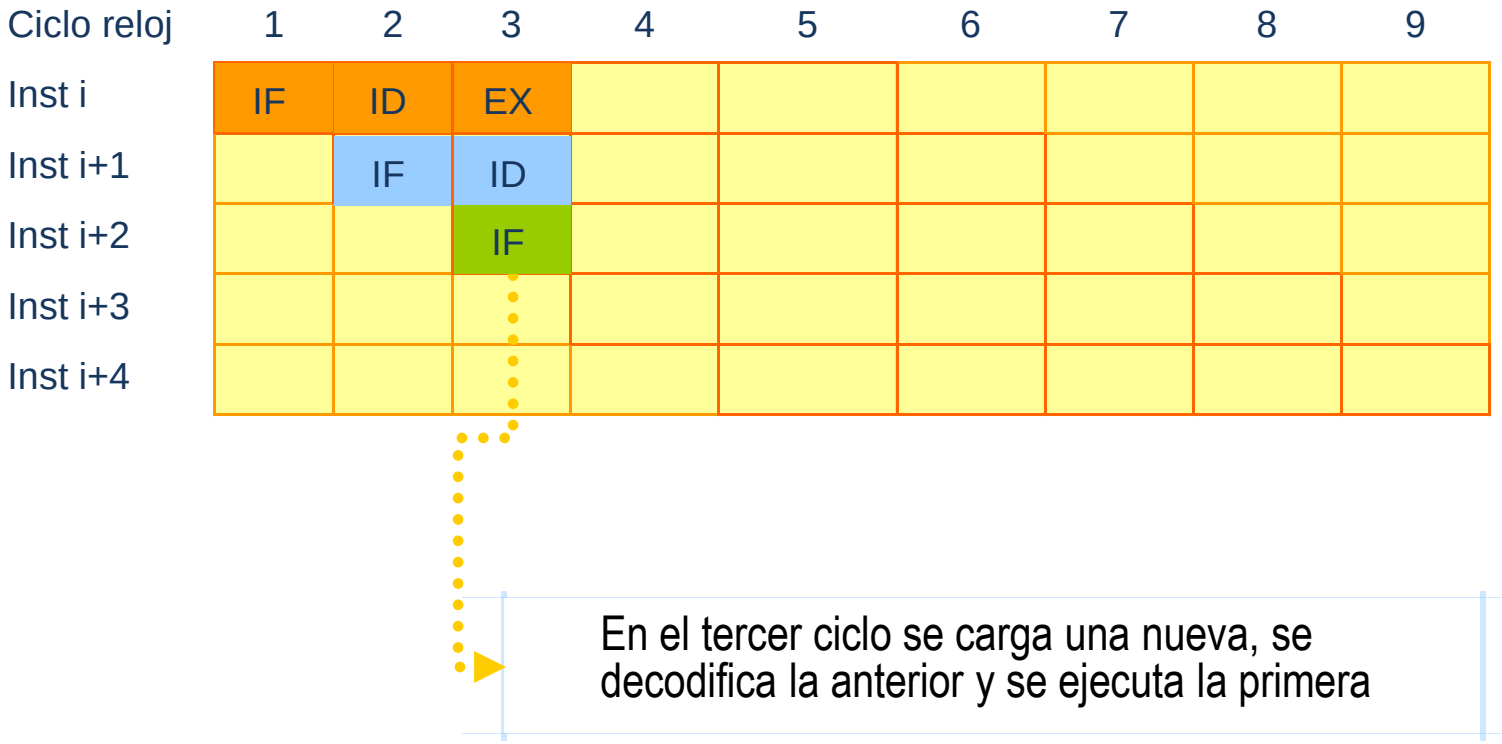
Segmentación de instrucciones

- La segmentación consiste en solapara la ejecución de instrucciones



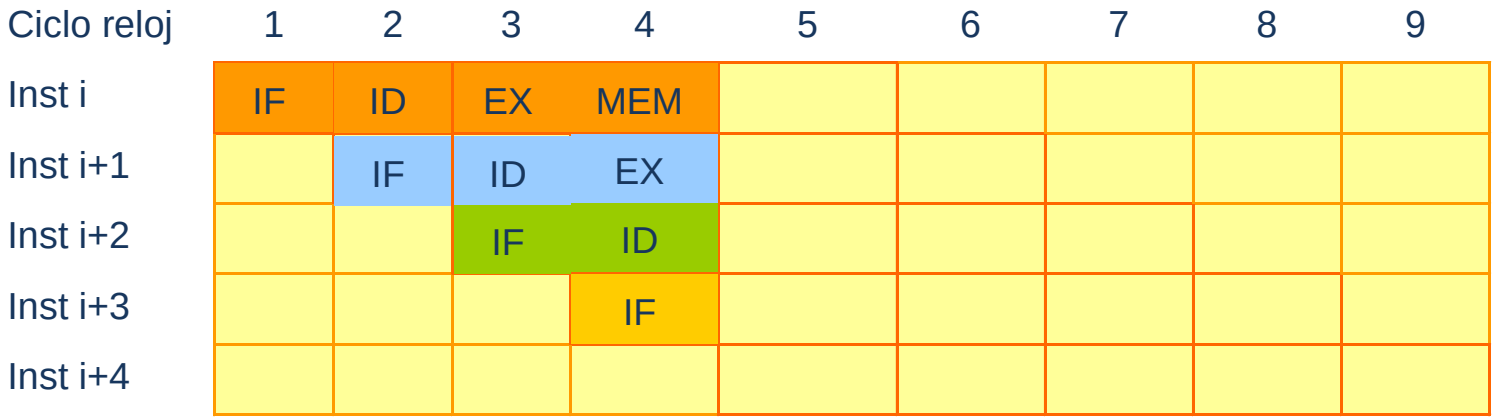
Segmentación de instrucciones

- La segmentación consiste en solapara la ejecución de instrucciones



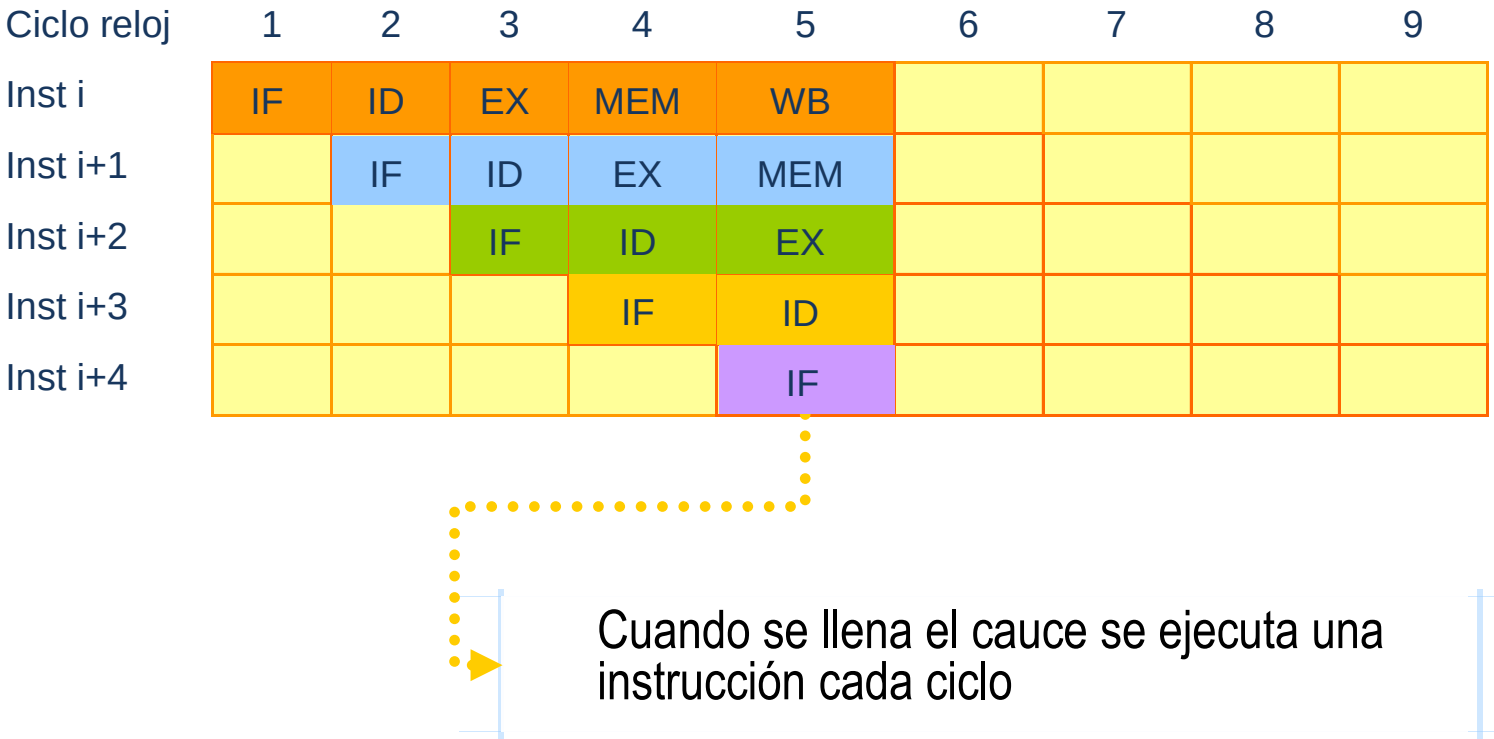
Segmentación de instrucciones

- La segmentación consiste en solapara la ejecución de instrucciones

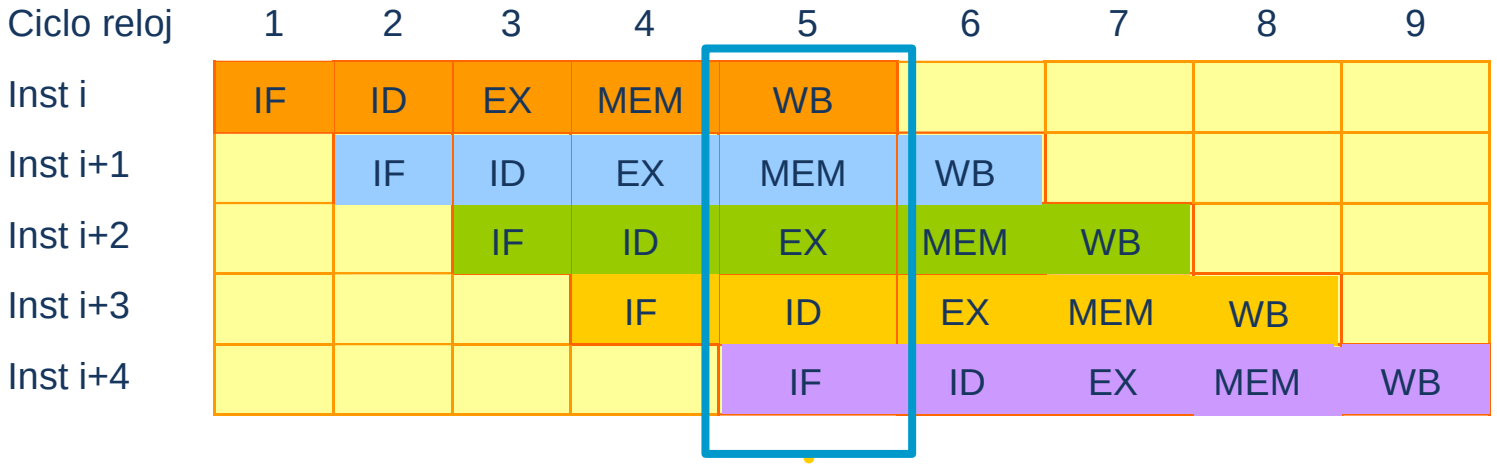


Segmentación de instrucciones

- La segmentación consiste en solapara la ejecución de instrucciones



Segmentación de instrucciones



- Cada paso constituye una etapa de la segmentación.
- Cada ciclo, cinco instrucciones en ejecución
- La segmentación incrementa la **productividad** sin reducir el tiempo de ejecución de una instrucción individual.
- Los programas se ejecutan más rápido

Ejemplo rendimiento

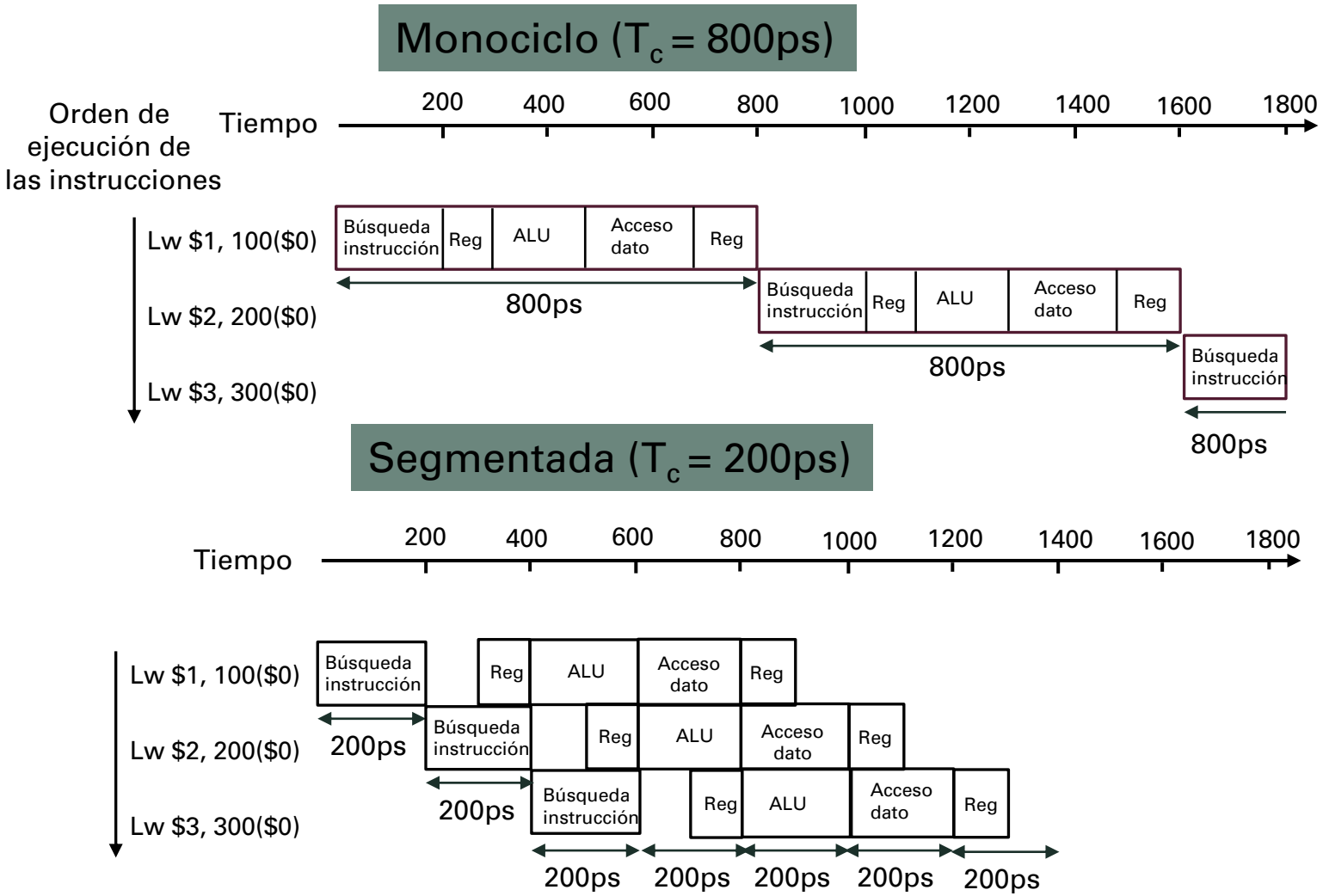
- Considerar los siguientes tiempos de operación para los elementos funcionales superiores de la ruta de datos:
  - 100ps para lectura o escritura en registros
  - 200ps para el acceso a memoria y operaciones ALU
  - El retardo del resto de elementos es despreciable
- Compara la ruta de datos monociclo con la segmentada

Tipo Instr.	Unidades funcionales utilizadas					Total
Tipo - R	Buscar Instr. (200ps)	Acceso Reg. (100ps)	ALU (200ps)	Acceso Reg. (100ps)		600ps
Lw	Buscar Instr. (200ps)	Acceso Reg. (100ps)	ALU (200ps)	Acceso memoria (200ps)	Acceso Reg. (100ps)	800ps
Sw	Buscar Instr. (200ps)	Acceso Reg. (100ps)	ALU (200ps)	Acceso memoria (200pss)		700ps
Salto Cond.	Buscar Instr. (200ps)	Acceso Reg. (100ps)	ALU (200ps)			500ps

Monociclo ($T_c = 800ps$)

Segmentada ($T_c = 200ps$)

Ejemplo rendimiento



Ejemplo rendimiento

$$Aceleración = \frac{Tiempo\ medio_{\text{sin } segmentación}}{Tiempo\ medio_{con\ segmentación}} = \frac{800}{200} = 4$$

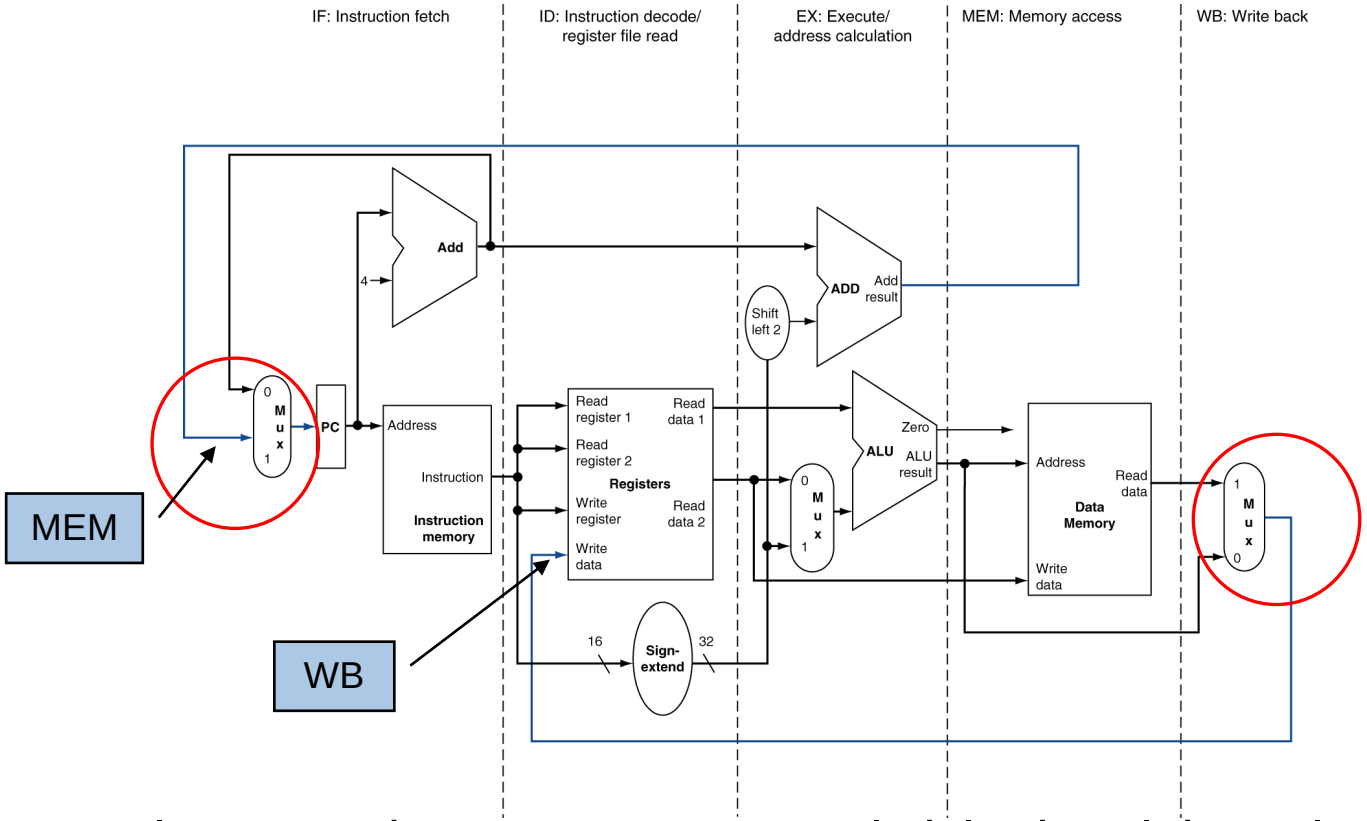
La ruta de datos segmentada es 4 veces más rápida

Ganancia en la segmentación

- Si todas las etapas están balanceadas
  - Es decir, tienen el mismo tiempo (caso ideal)
  - $Tiempo\ entre\ instrucciones_{segmentada} = \frac{Tiempo\ entre\ instrucciones_{\text{sin}\ segmentación}}{Número\ de\ etapas}$
  - En el caso ideal, la ganancia (aceleración) es aproximadamente igual al número de etapas de la segmentación
- Si no están balanceadas, la aceleración es menor
- La aceleración se debe a que se incrementa la productividad
  - La latencia (tiempo para ejecutar cada instrucción) no decrece

Segmentación de la ruta de datos

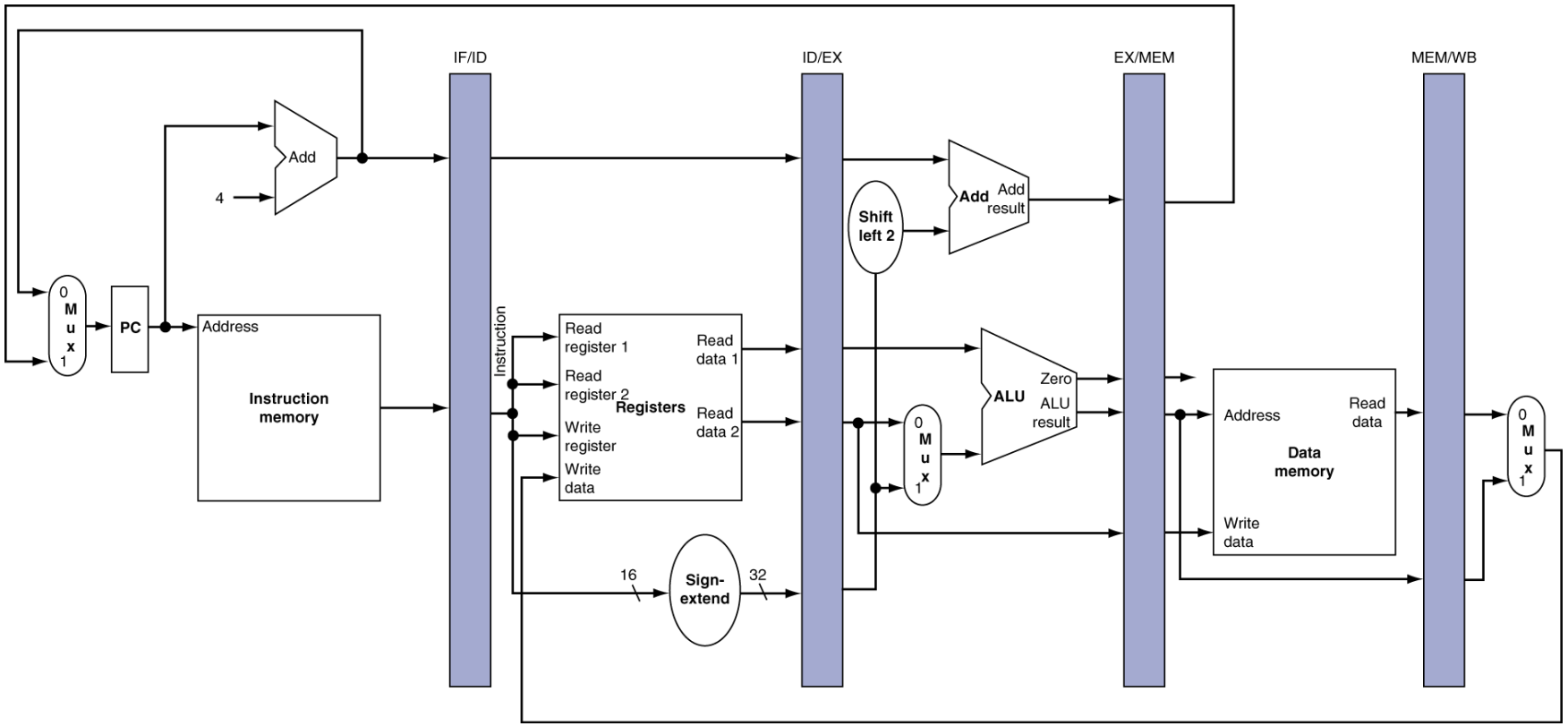
- Evolución de una instrucción en la ruta de datos



- El PC y registros aparecen en el ciclo de reloj en el que son leídos
- Indicamos las escrituras en la etapa mediante multiplexores

Registros de segmentación

- Es necesario incluir registros entre las etapas
  - Para mantener la información obtenida en el ciclo anterior

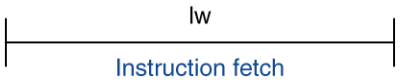


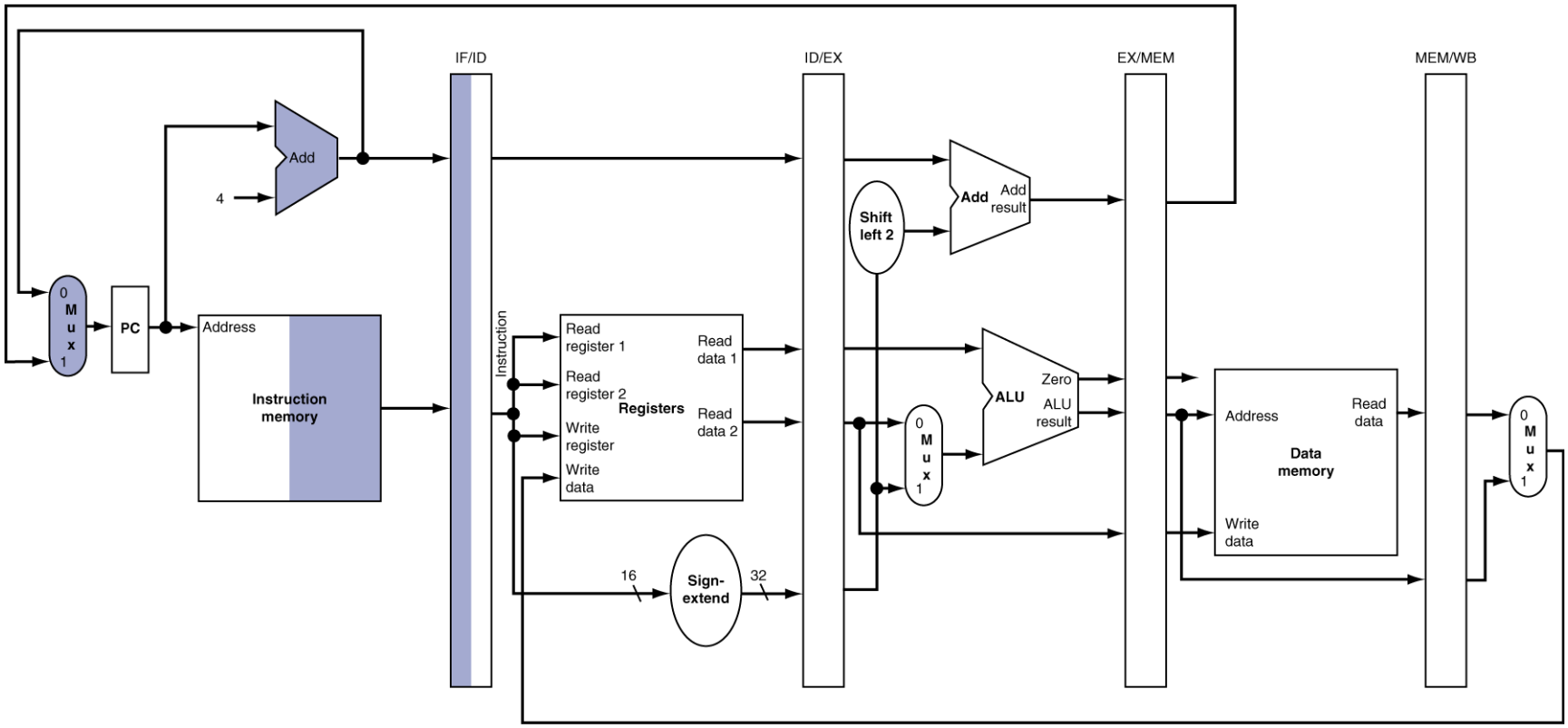
Operación de la segmentación

- Mostraremos el flujo de las instrucciones a través de la ruta de datos segmentada ciclo a ciclo
  - Diagrama de segmentación de un único ciclo de reloj
    - Muestra el uso de la ruta en un único ciclo de reloj
    - Se resalta los recursos utilizados
  - Posteriormente mostraremos el diagrama multiciclo
    - Grafo de operación a lo largo del tiempo
- Mostraremos el diagrama de segmentación de un ciclo para la instrucción lw ya que estará activa en cada una de las cinco etapas.

IF: Búsqueda de la instrucción

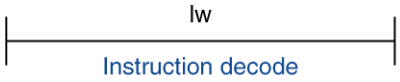
- Para todas las instrucciones
- La etapa ocurre antes de identificarse la instrucción: se guarda la información en el registro de segmentación IF/ID

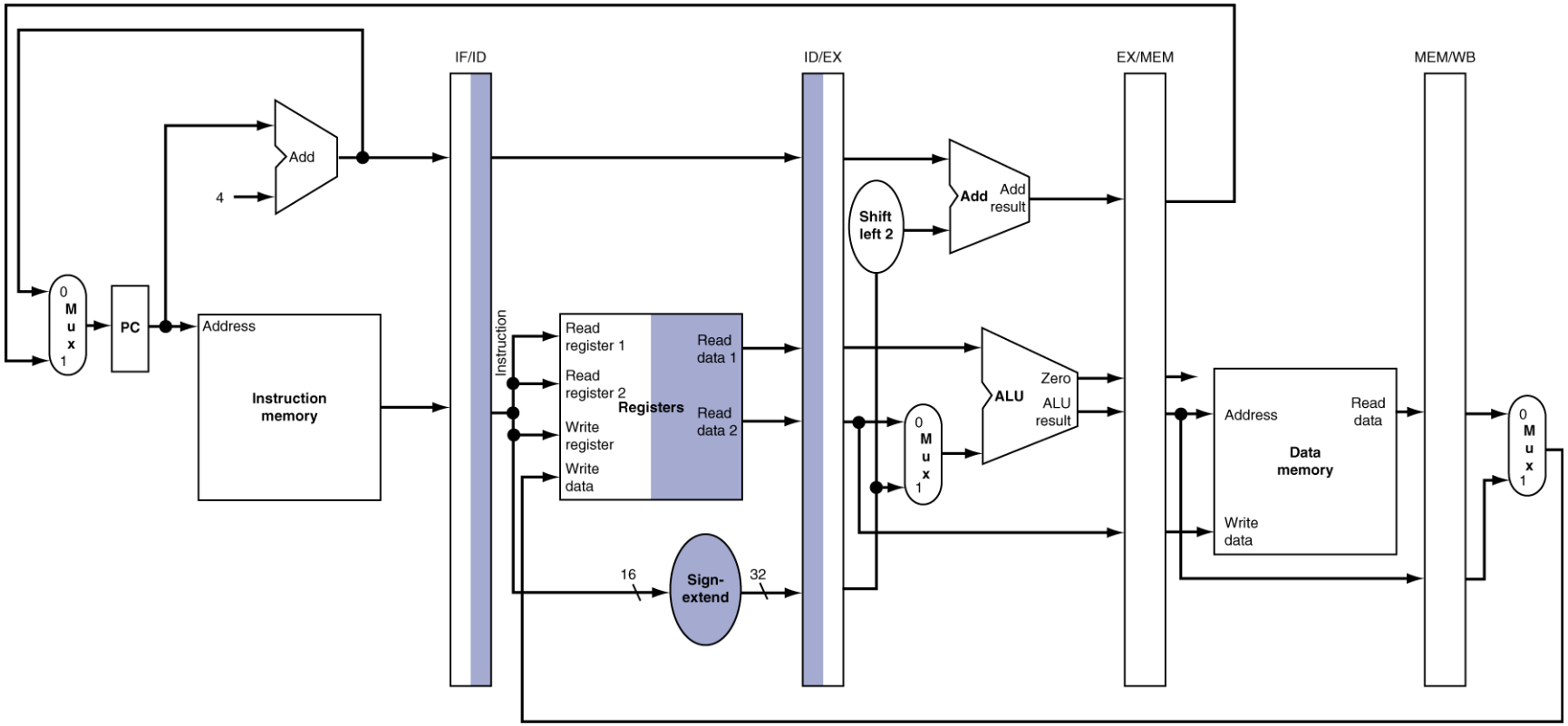




ID: Decodificación de la instrucción

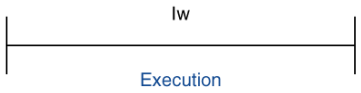
- Para todas las instrucciones
- Decodificación de la instrucción y lectura de los registros. Se guarda toda la información en ID/EX que pueda necesitar cualquier instrucción en ciclos de reloj posteriores

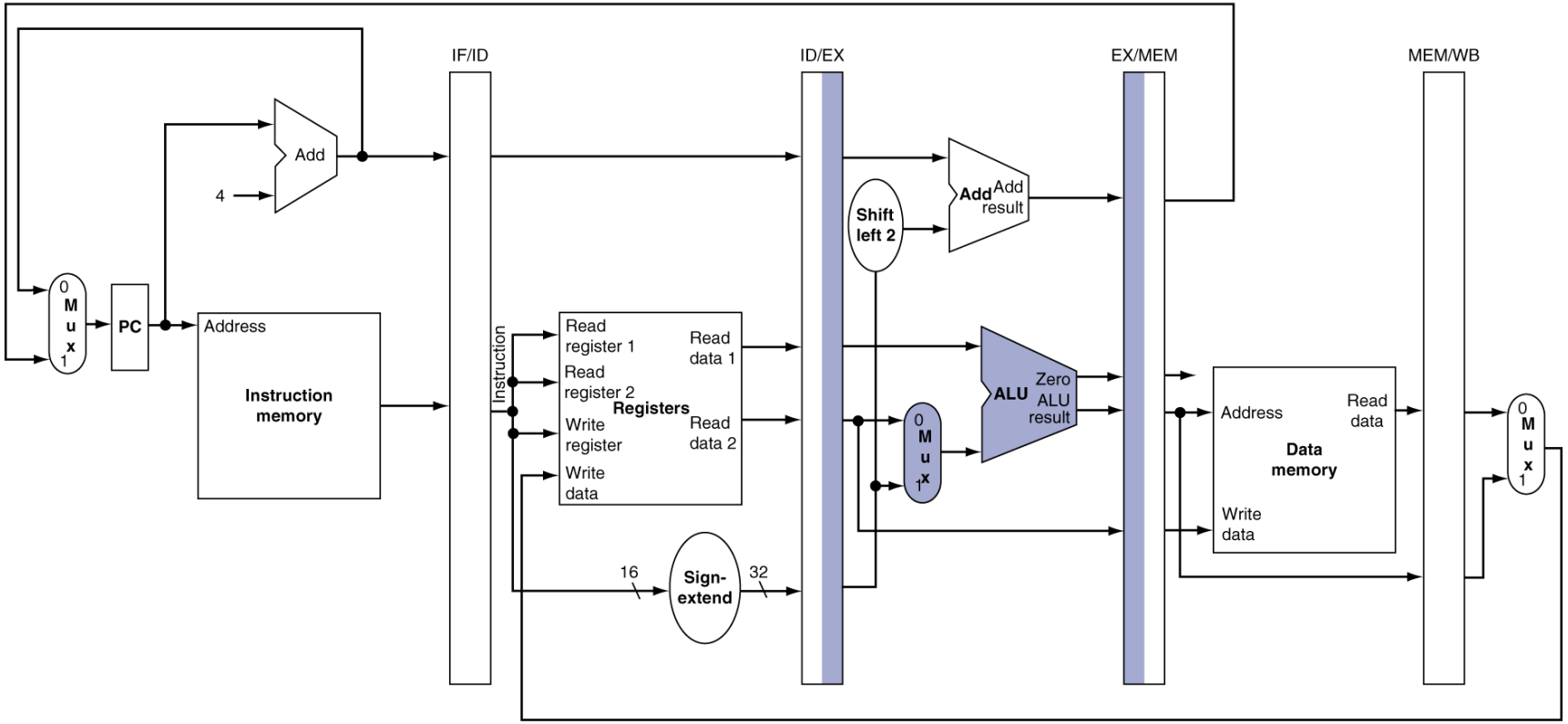




EX: Ejecución y cálculo de direcciones

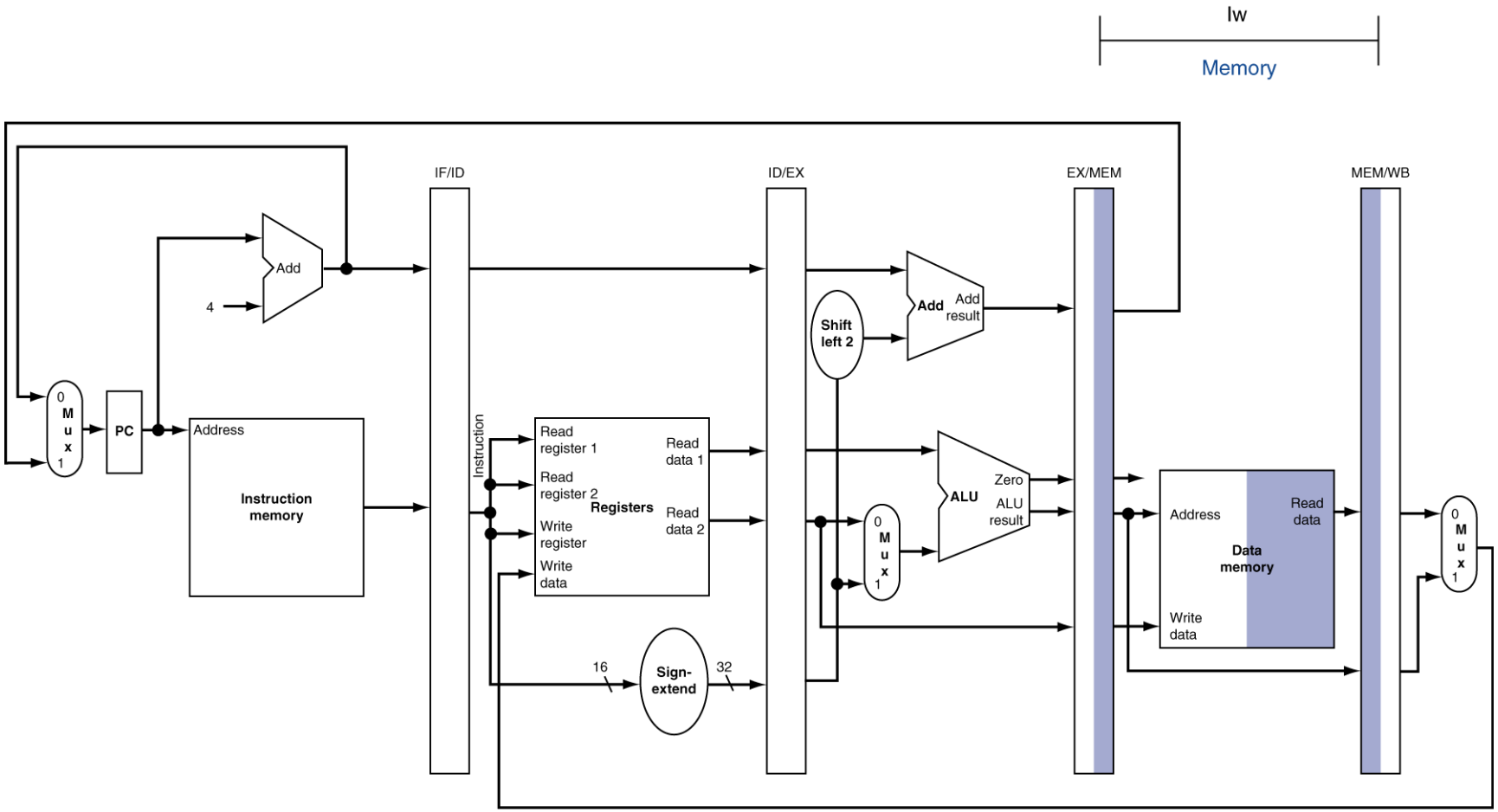
- Calcula dirección de memoria y la guarda en el registro EX/MEM:
  - Lee el registro 1 y la extensión de signo del desplazamiento guardado en el registro de segmentación ID/EX y los suma utilizando la ALU





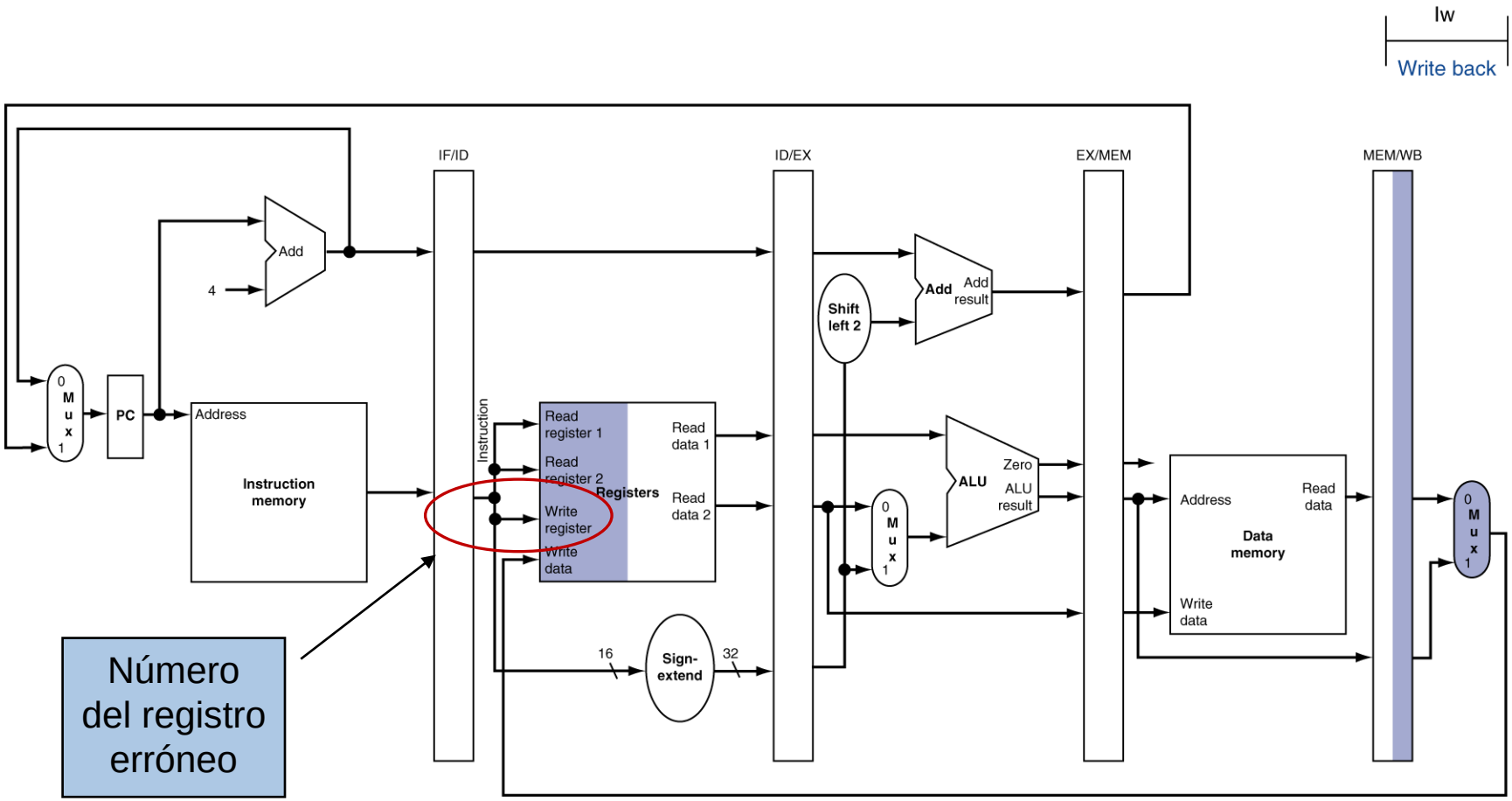
MEM: Acceso a memoria

- Lee el dato en memoria utilizando la dirección guardada en el registro de segmentación EX/MEM y guarda el dato leído en el registro MEM/WB



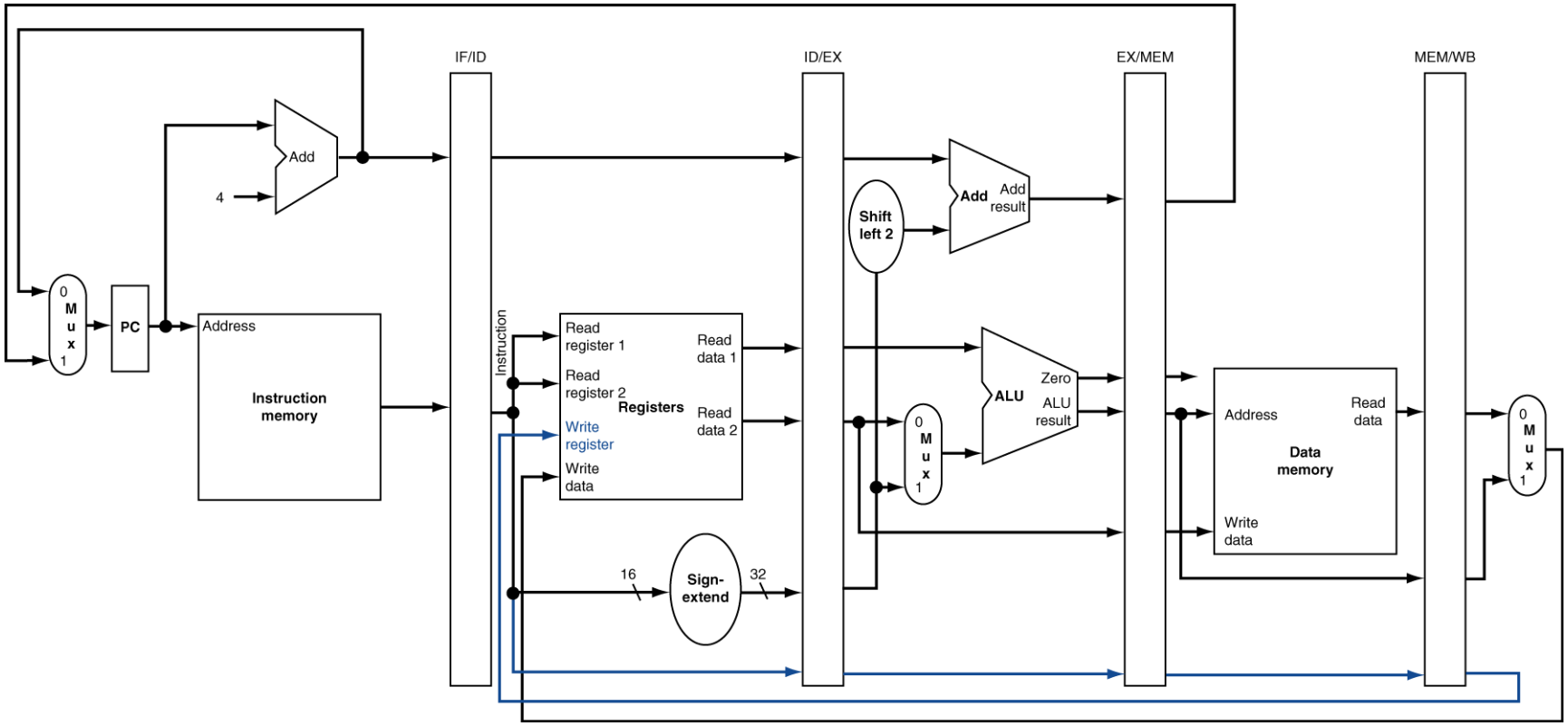
WB: Escritura en registros

- Lee el dato guardado en el registro de segmentación MEM/WB y lo escribe en registro correspondiente
  - Es necesario preservar el registro destino a través de los registros de segmentación



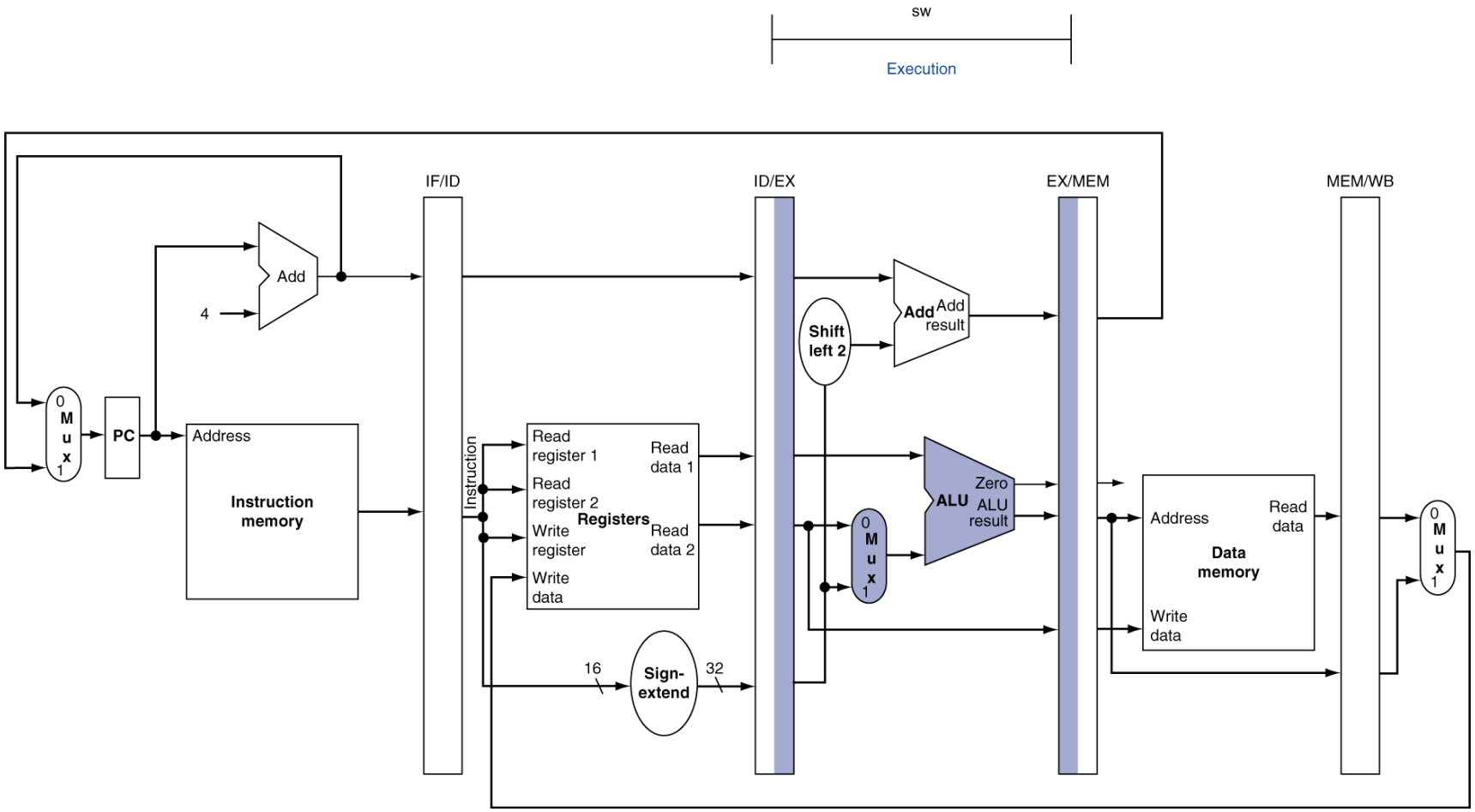
Ruta de datos corregida

- El registro de escritura guardado en IF/ID se pasa a través de todos los registros de segmentación hasta ser utilizado en la etapa WB



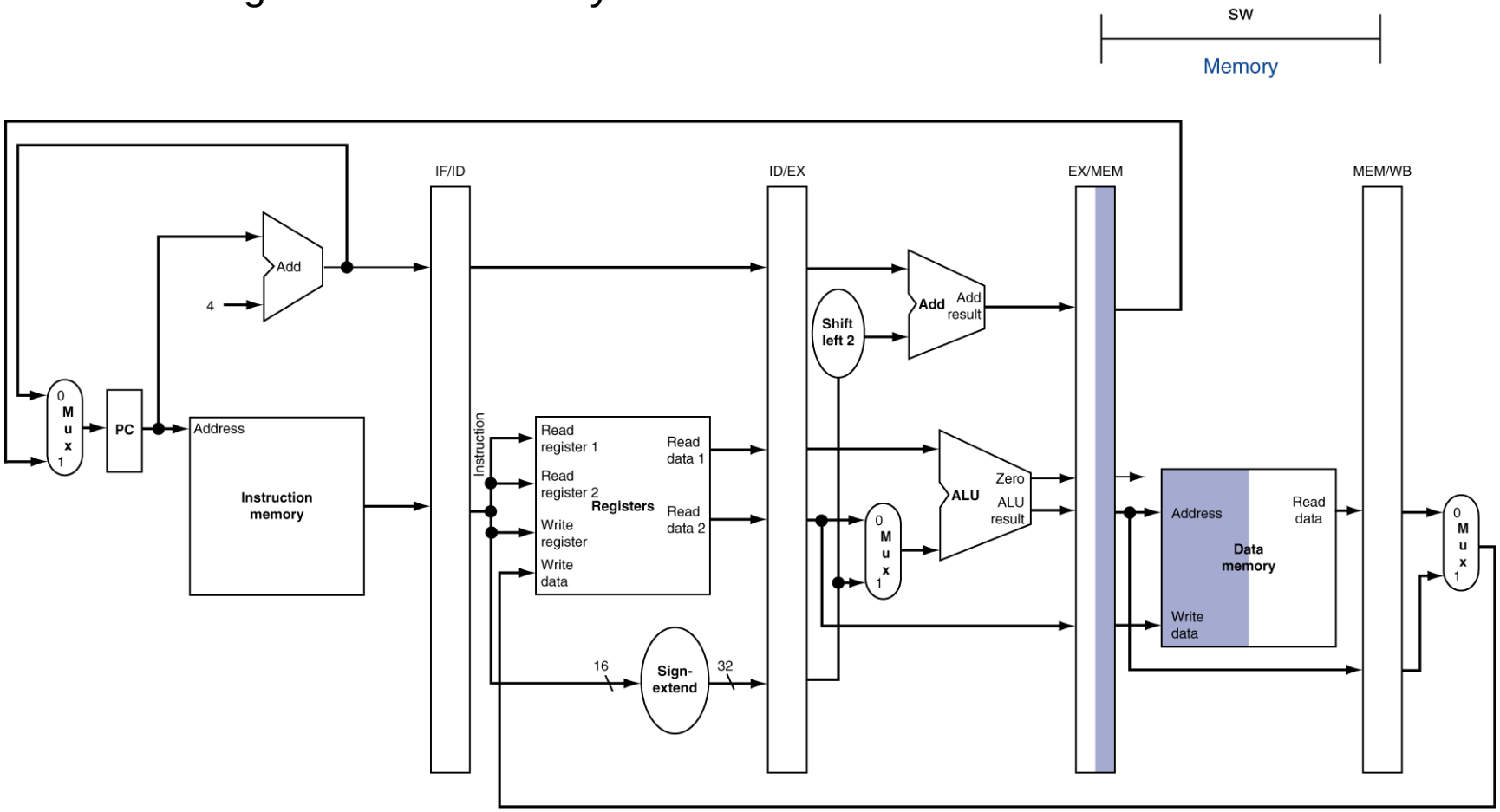
Etapa EX para la instrucción sw

- Calcula dirección de memoria y la guarda en el registro EX/MEM



Etapa MEM para la instrucción sw

- Almacena el dato en memoria:
  - El dato leído del registro ha pasado por los registros de segmentación ID/EX y EX/MEM



Etapa WB para la instrucción sw

- No ocurre nada en esta etapa. La instrucción la atraviesa sin hacer nada.

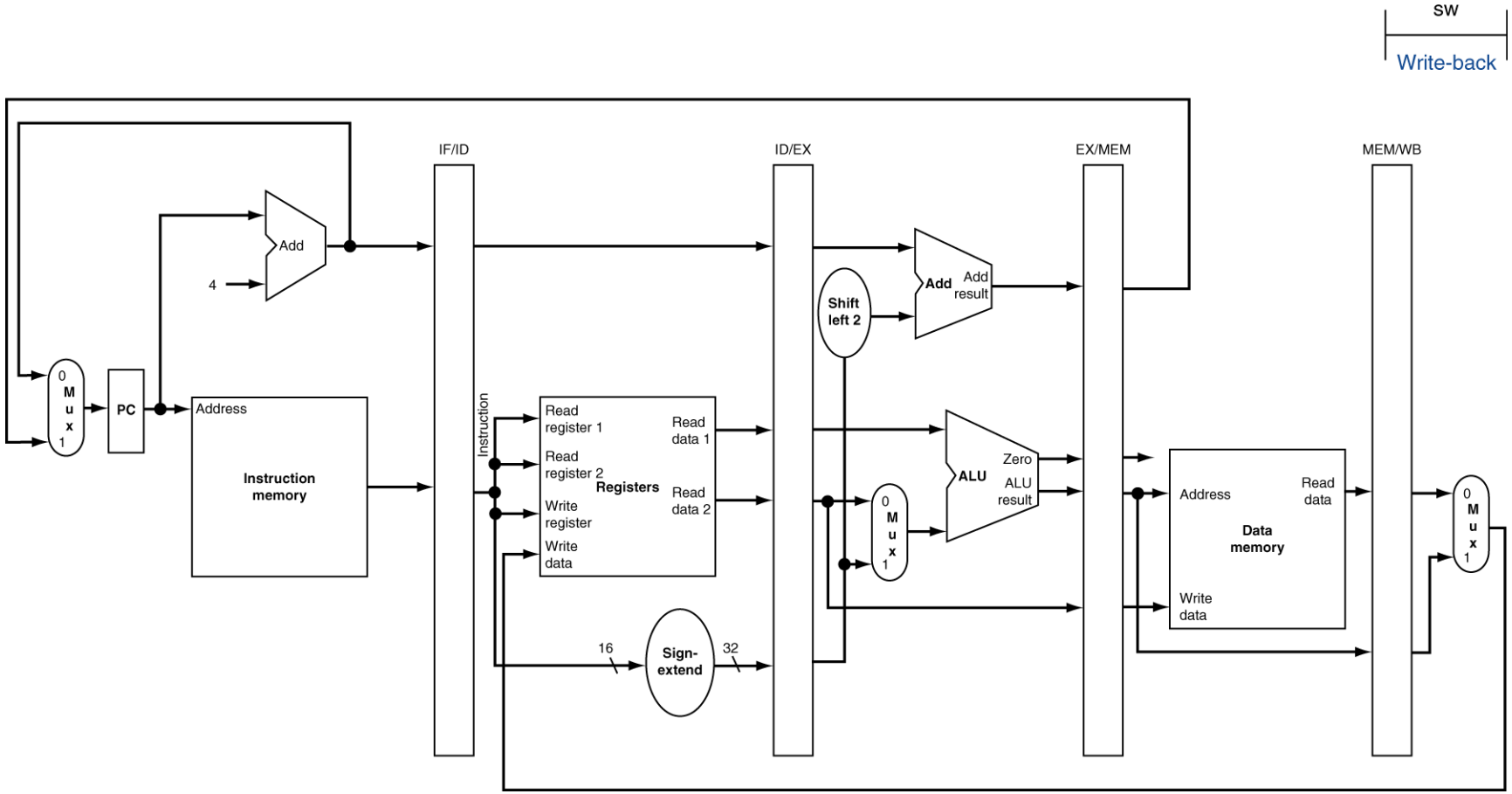
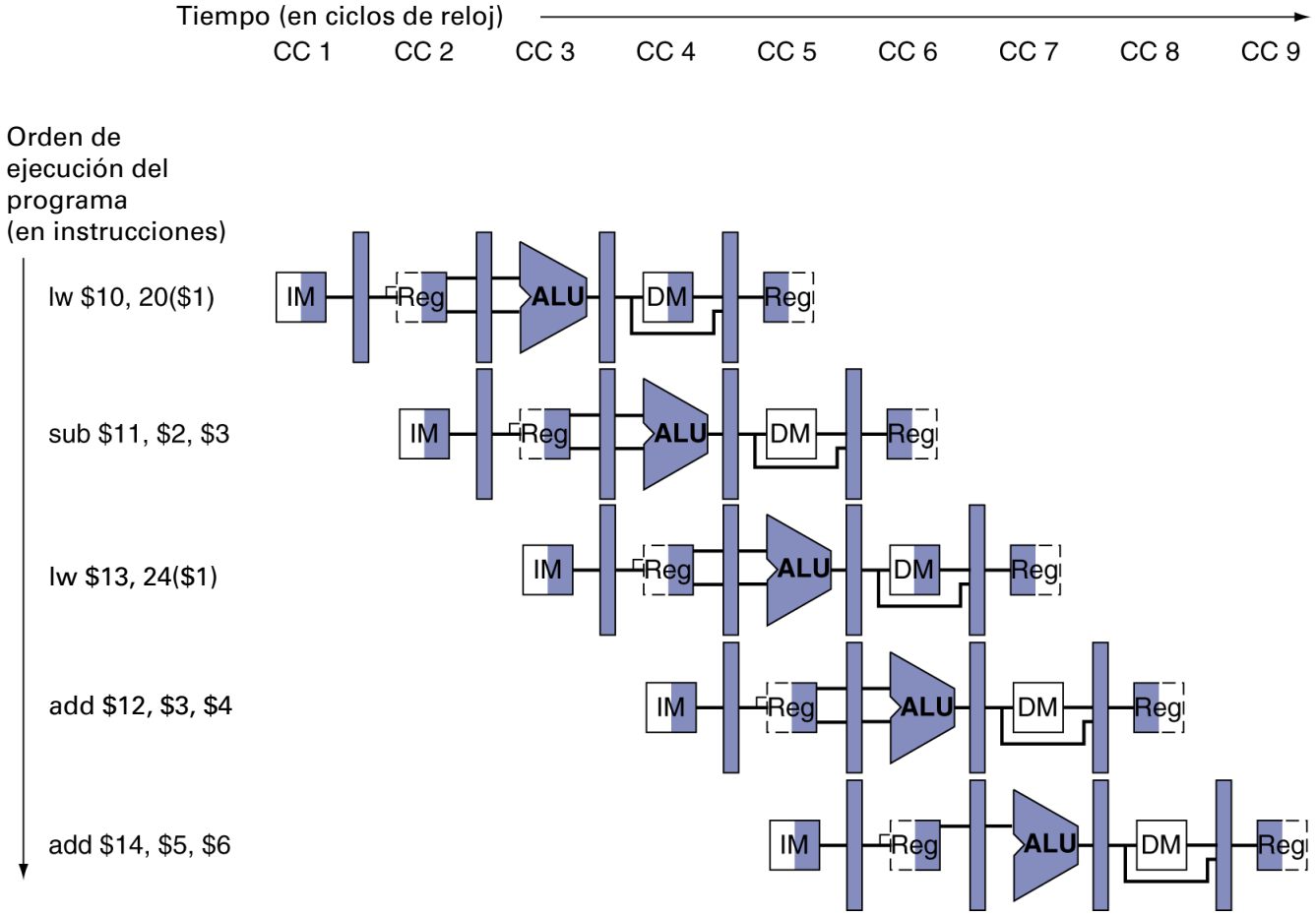


Diagrama de segmentación multiciclo

- Representa gráficamente las cinco etapas de la ruta de datos segmentada mostrando el recurso utilizado en cada ciclo de reloj para cada instrucción.



Versión tradicional

- Diagrama de segmentación multiciclo tradicional, etiqueta con el nombre cada etapa utilizada en cada ciclo de reloj para cada instrucción en ejecución.

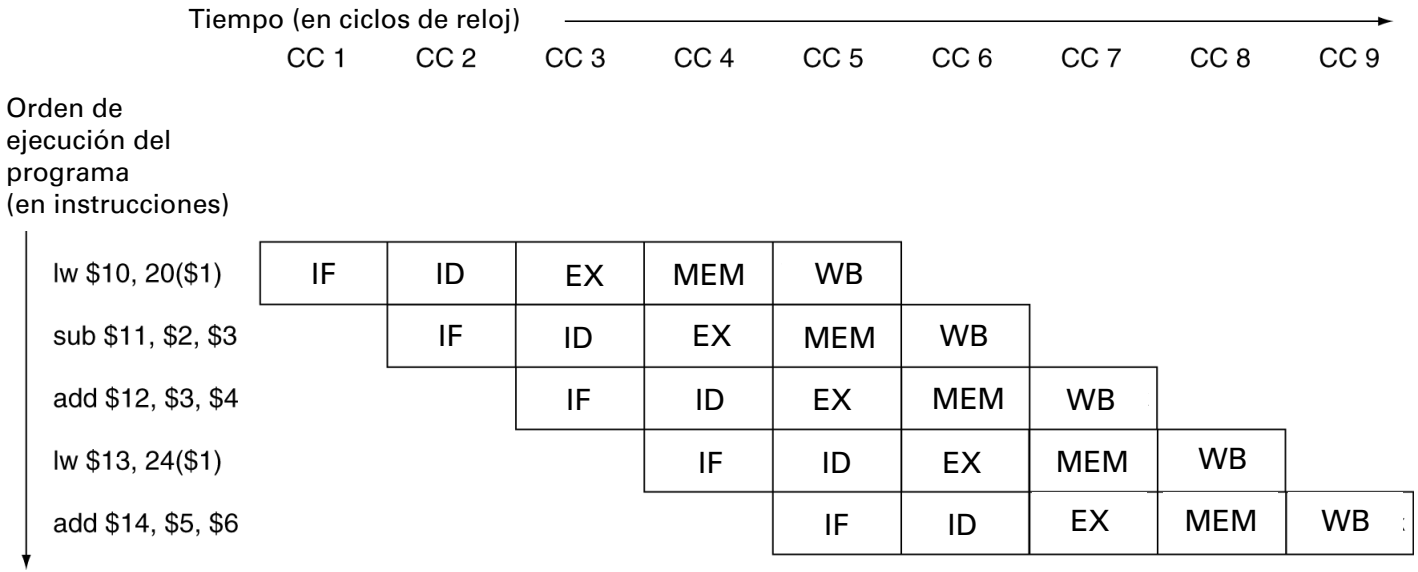
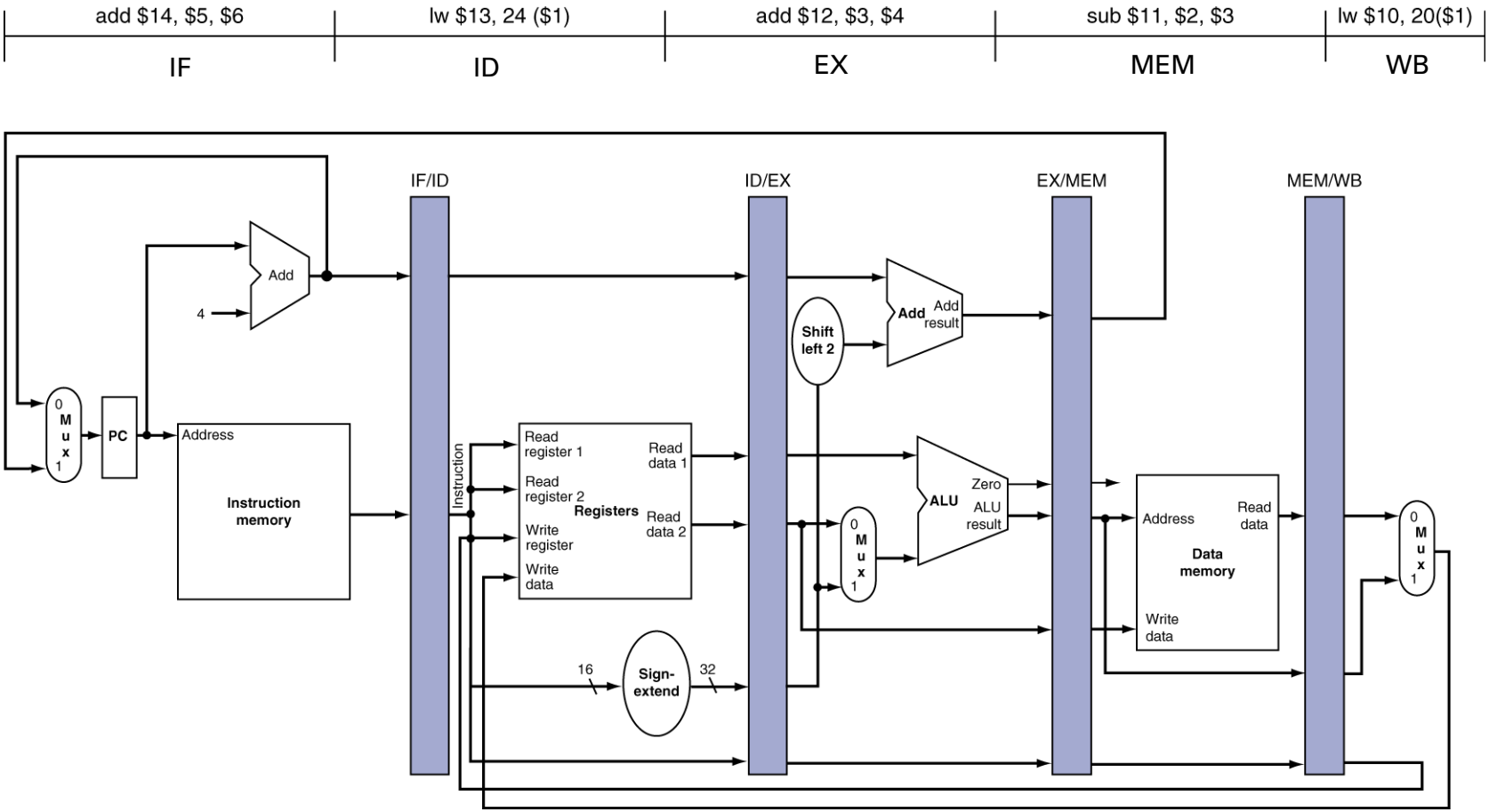


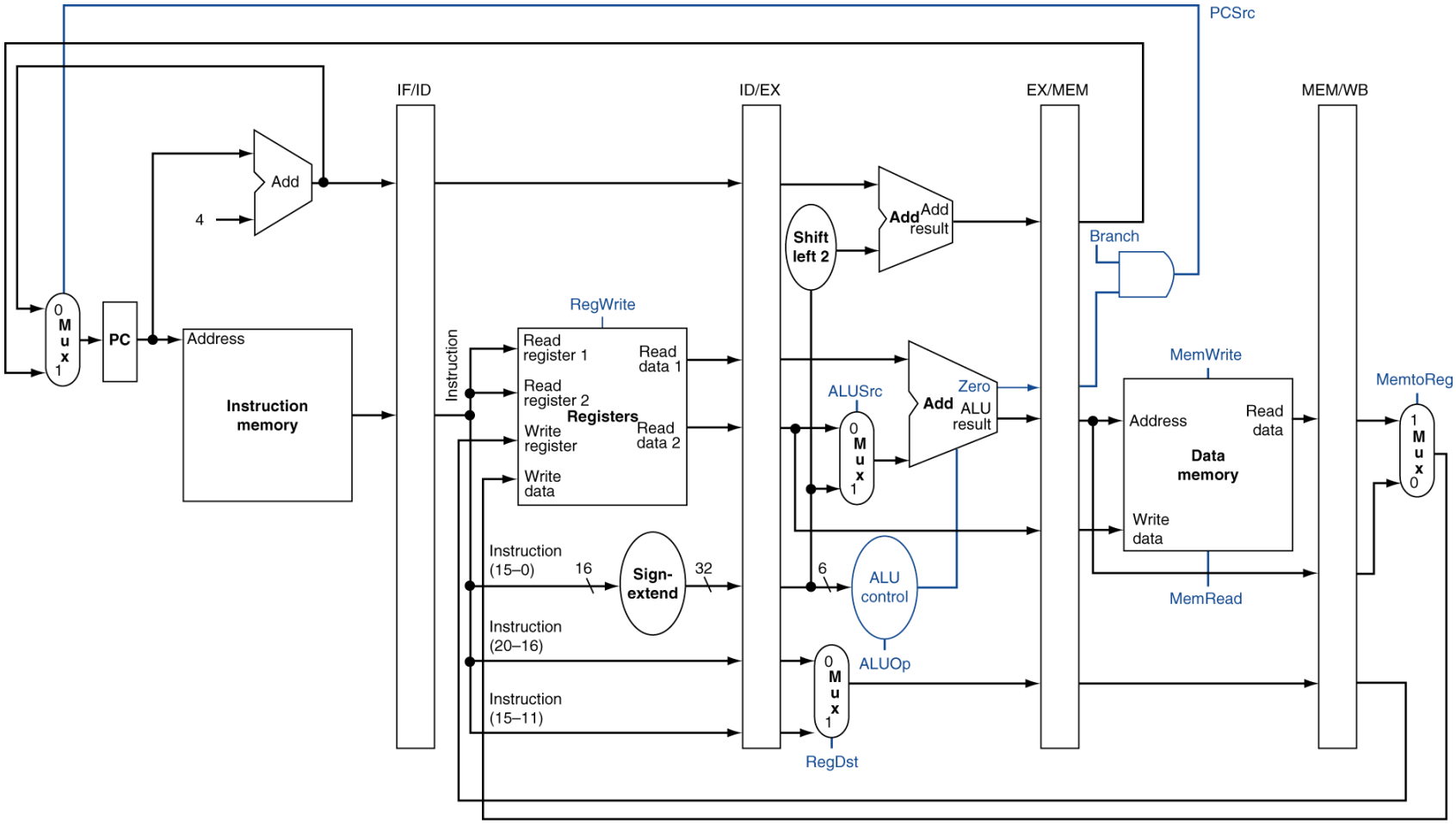
Diagrama de segmentación uniciclo

- Muestra el estado de toda la ruta de datos segmentada durante **un ciclo de reloj**. Indica la tapa en la que se encuentra cada instrucción.



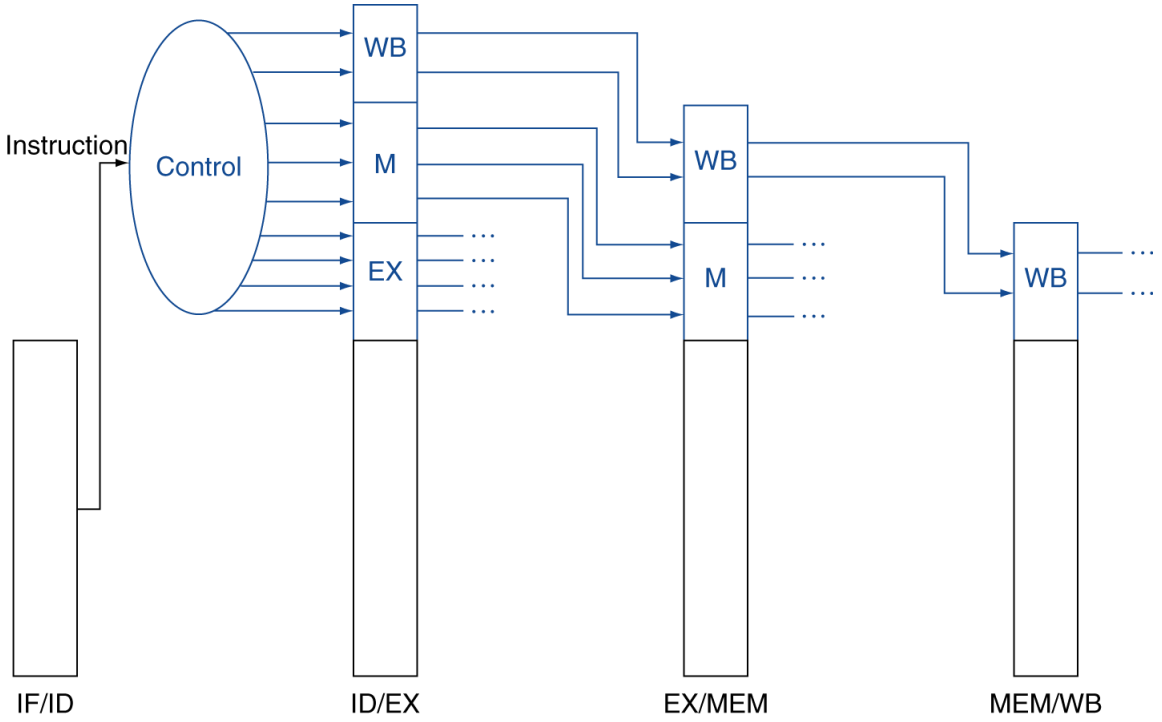
Identificar señales de control

- Los registros PC, IF/ID, ID/EX, EX/MEM y MEM/WB se escriben en todos los ciclos de reloj, por tanto, no tienen señales de control separadas.

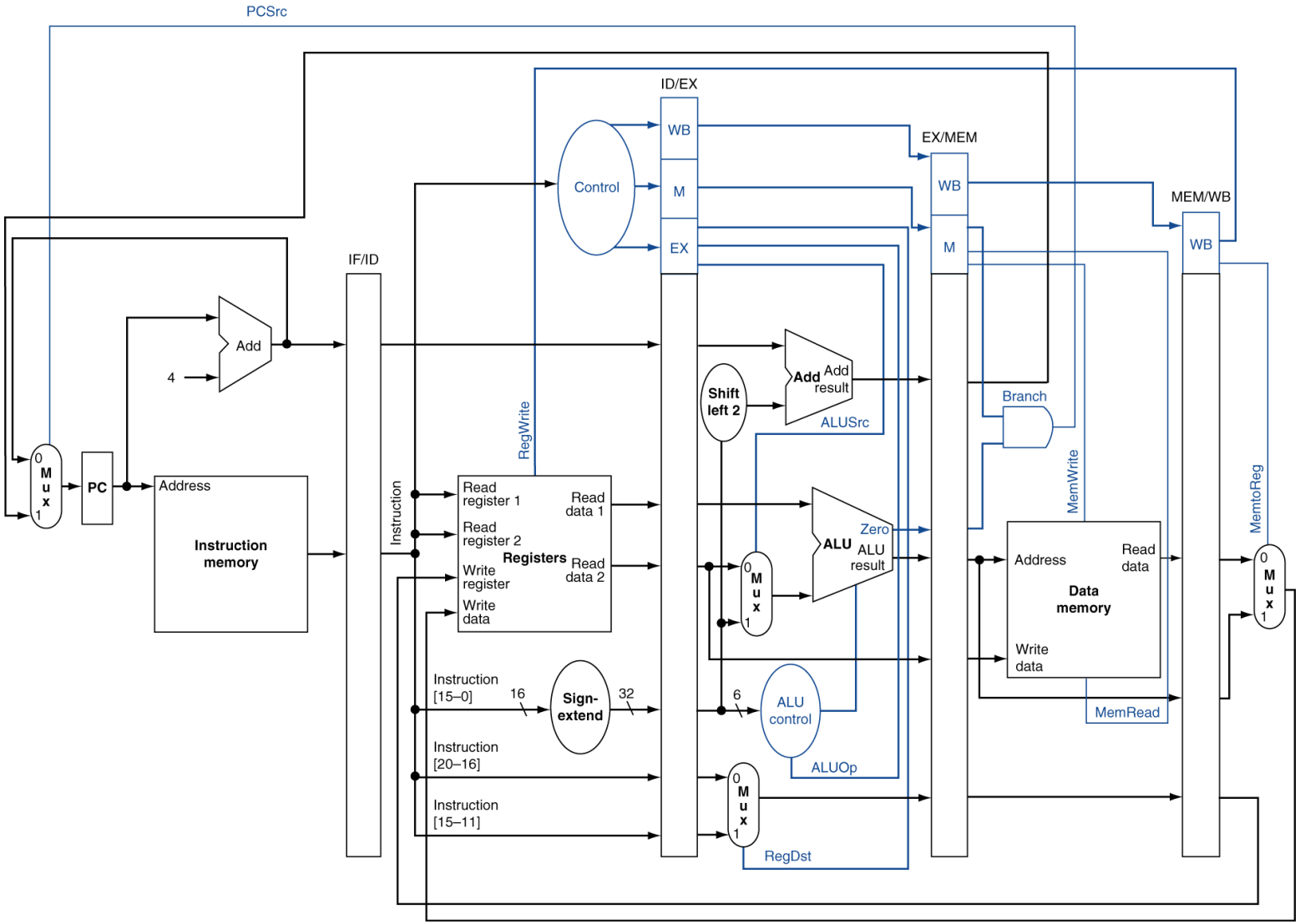


Control de la segmentación

- Se extienden los registros de segmentación para incluir el valor de las señales de control.
- La información de control se crea en la etapa de decodificación para cada instrucción.
- Las señales de control se usan en la etapa apropiada a medida que la instrucción se mueve a través de la segmentación.



Ruta de datos segmentada y control



3. Riesgos de la segmentación

- **Riesgos estructurales.** Surgen de conflictos de los recursos cuando el hardware no puede soportar todas las combinaciones de instrucciones en ejecución (p.ej. acceso a memoria, banco registros, etc). No hay en nuestra ruta de datos (hay memoria separada para instrucciones y para datos).
- **Riesgos por dependencias de datos.** Surgen cuando una instrucción depende de los resultados de una instrucción anterior.
- **Riesgos de control.** Surgen de la segmentación de los saltos y otras instrucciones que cambian el PC.

3. Riesgos de la segmentación

- Consecuencias:
  - Impiden que se ejecute la siguiente instrucción
  - Puede hacer necesario detener la segmentación
  - Cuando una instrucción se detiene, las instrucciones posteriores también lo hacen
  - Las detenciones disminuyen la ganancia con respecto a la ideal

Riesgos por dependencia de datos

Riesgos por dependencia de datos

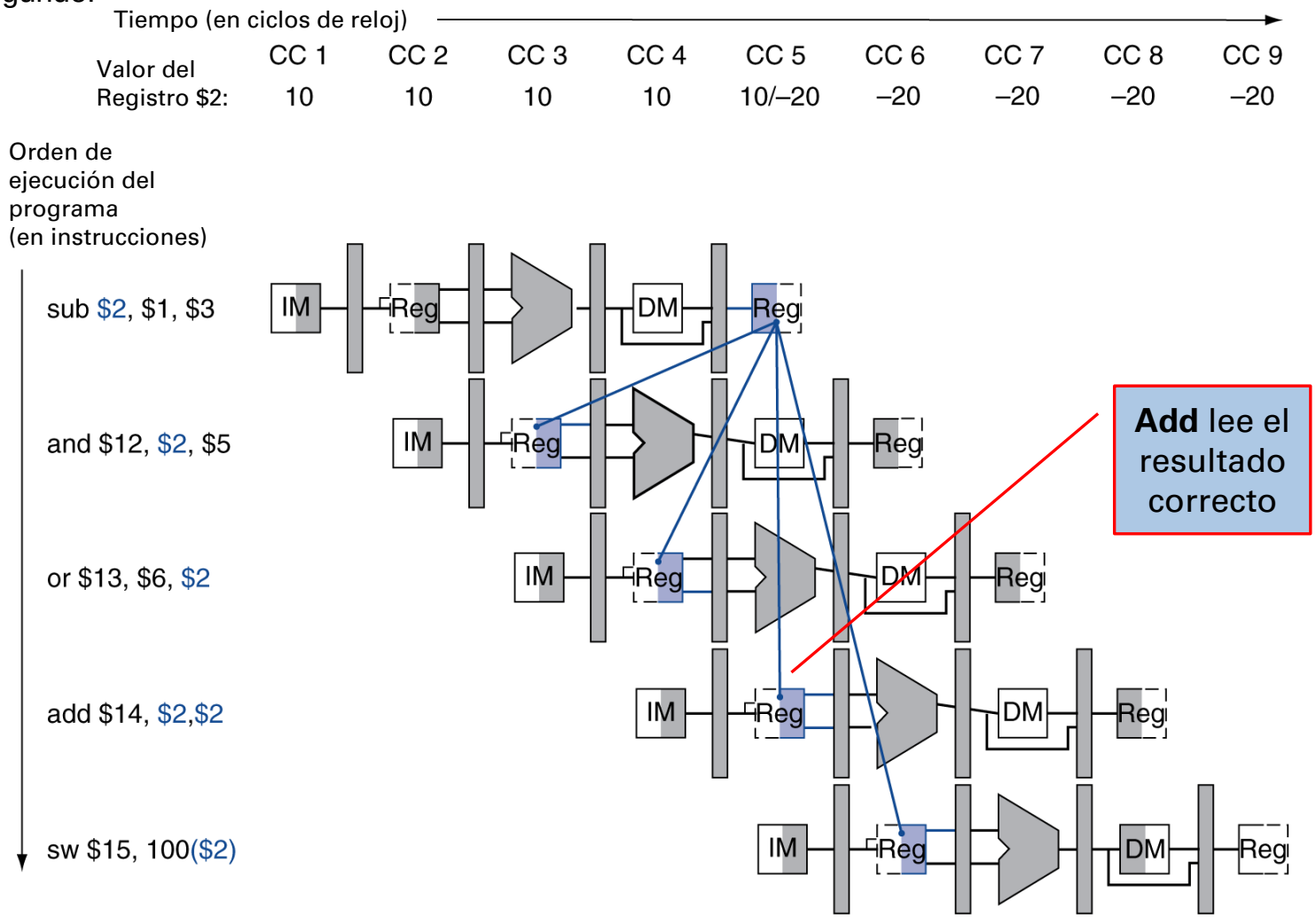
- Suponer la siguiente secuencia de instrucciones:

```
sub $2, $1,$3
and $12,$2,$5
or  $13,$6,$2
add $14,$2,$2
sw  $15,100($2)
```

- Las 4 últimas instrucciones dependen del resultado en el registro $2
- A no ser que se intervenga será necesario parar la segmentación

Riesgos por dependencia de datos

- Asumimos que la escritura en registros se realiza en el primer semiciclo de reloj y la lectura en el segundo.

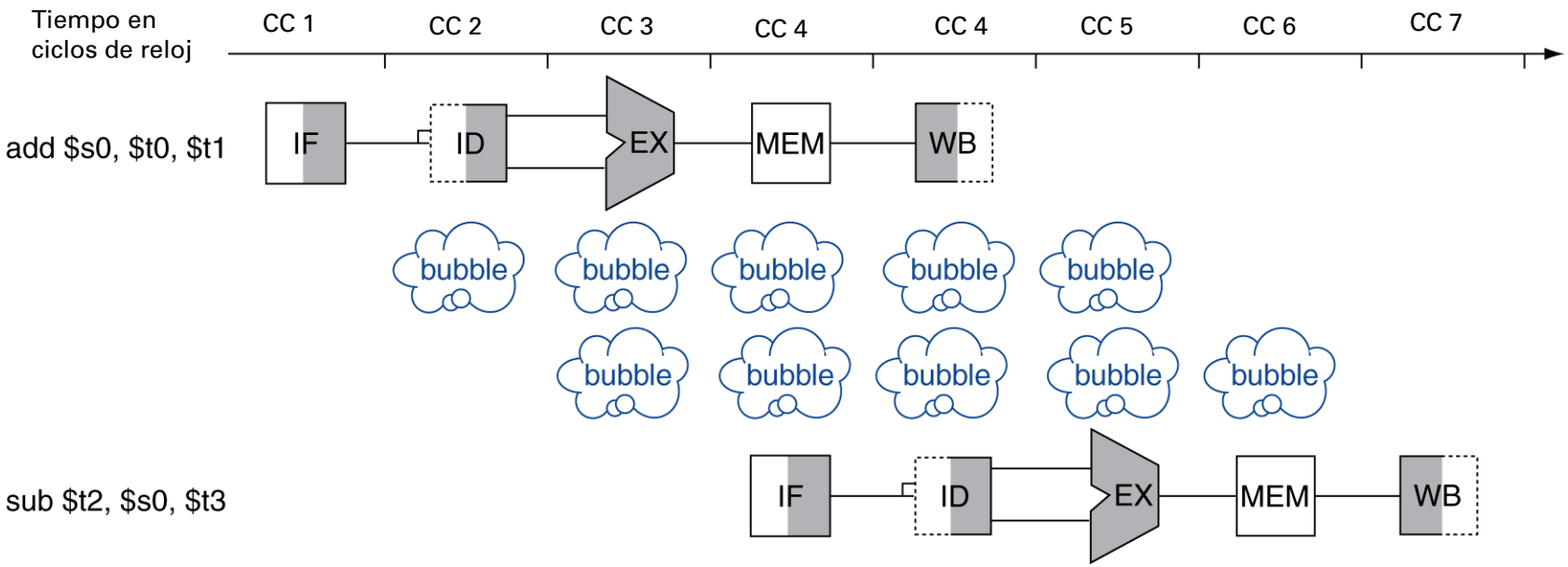


Riesgos por dependencia de datos

- La solución más sencilla es detener la segmentación hasta que se resuelve el riesgo
- A las detenciones se les llama burbujas ya que flotan en el cauce ocupando espacio, pero sin realizar trabajo.

Add $s0, $t0, $t1

Sub $t0, $s0, $t3



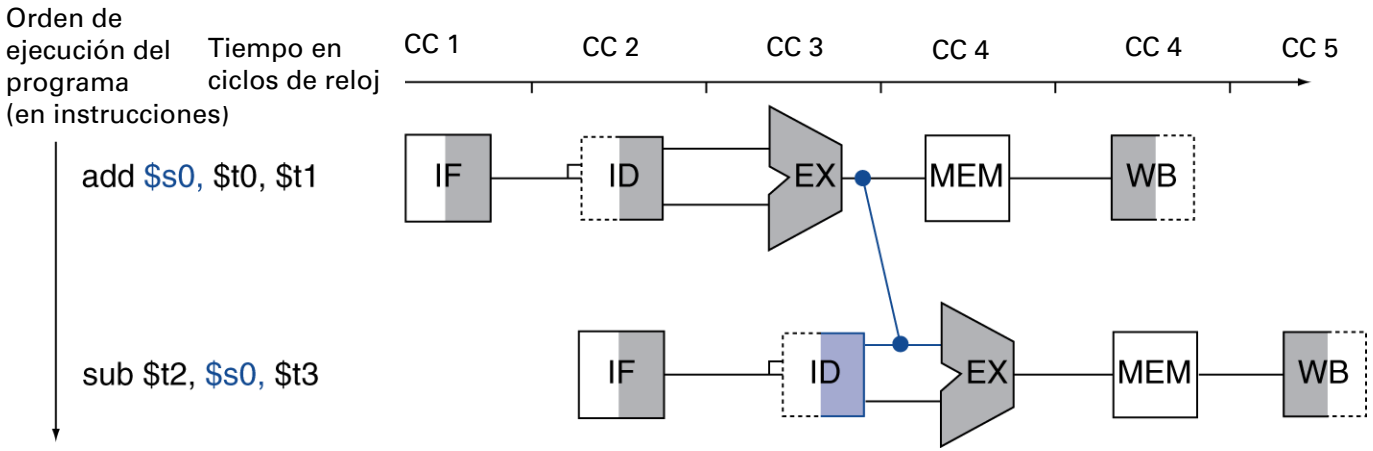
La instrucción *sub* no termina su ejecución hasta el ciclo de reloj 7

La técnica del adelantamiento

- Forwarding, bypassing
- Utilizar el resultado cuando se calcula

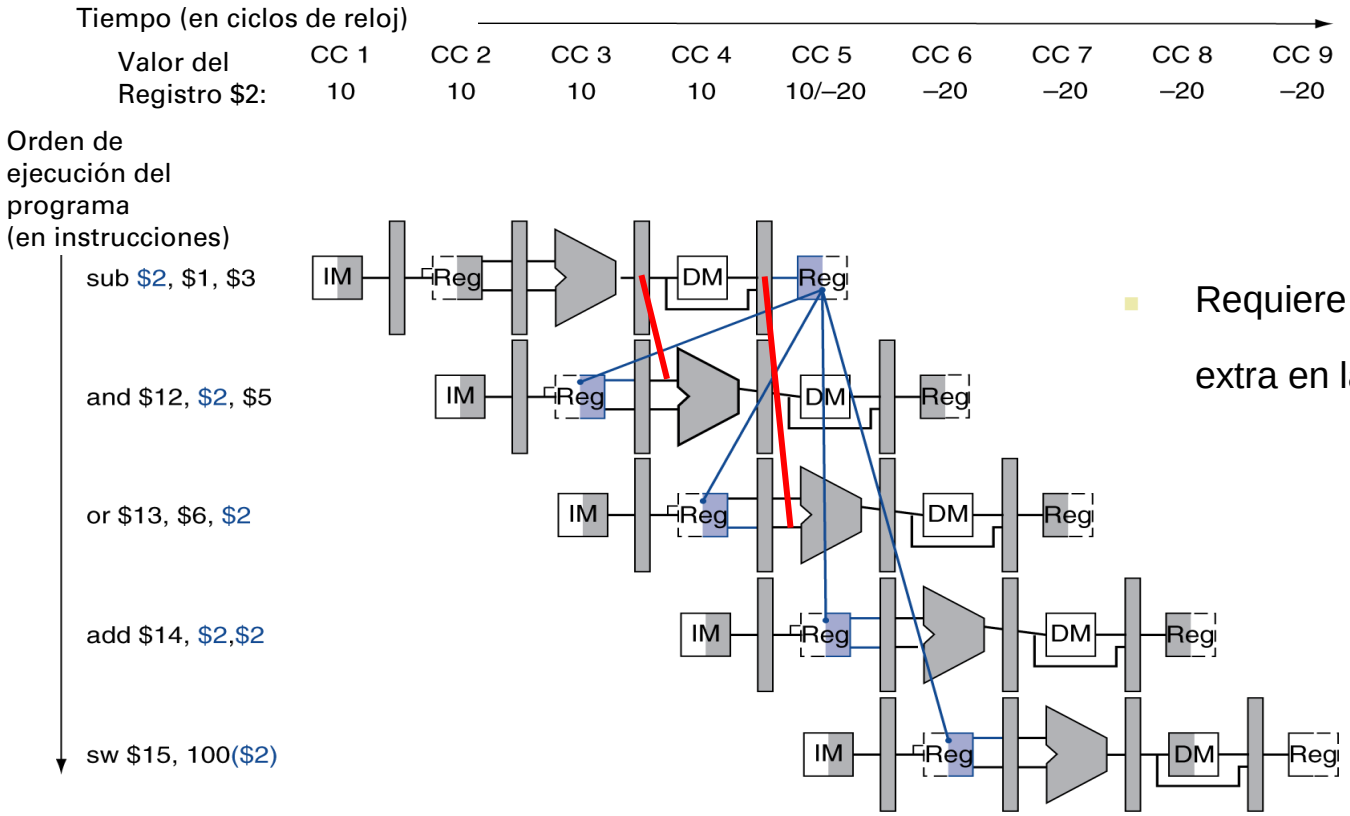
  Add $s0, $t0, $t1

  Sub $t0, $s0, $t3
- El resultado no se necesita realmente en la instrucción SUB hasta después de que la ADD realmente los produce, no es necesario esperar a que se guarde en el registro



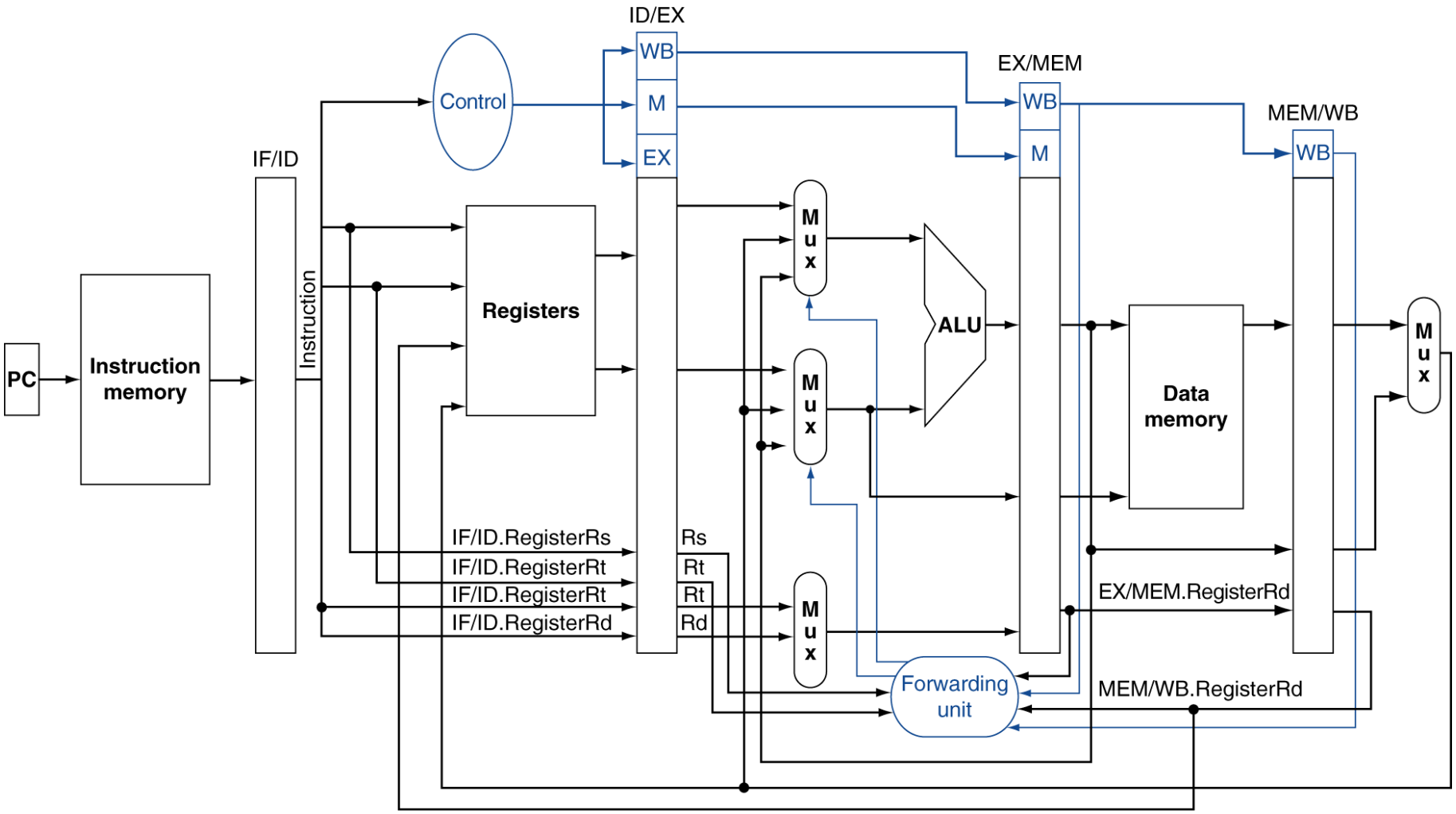
La técnica del adelantamiento

- Las entradas para *and* y *or* se adelantan desde los registros EX/MEM y MEM/WB
- Si el hardware de adelantamiento detecta que el registro destino de la instrucción es fuente de la operación ALU en curso, la lógica de control selecciona el valor adelantado como entrada a la ALU en lugar del valor leído en los registros



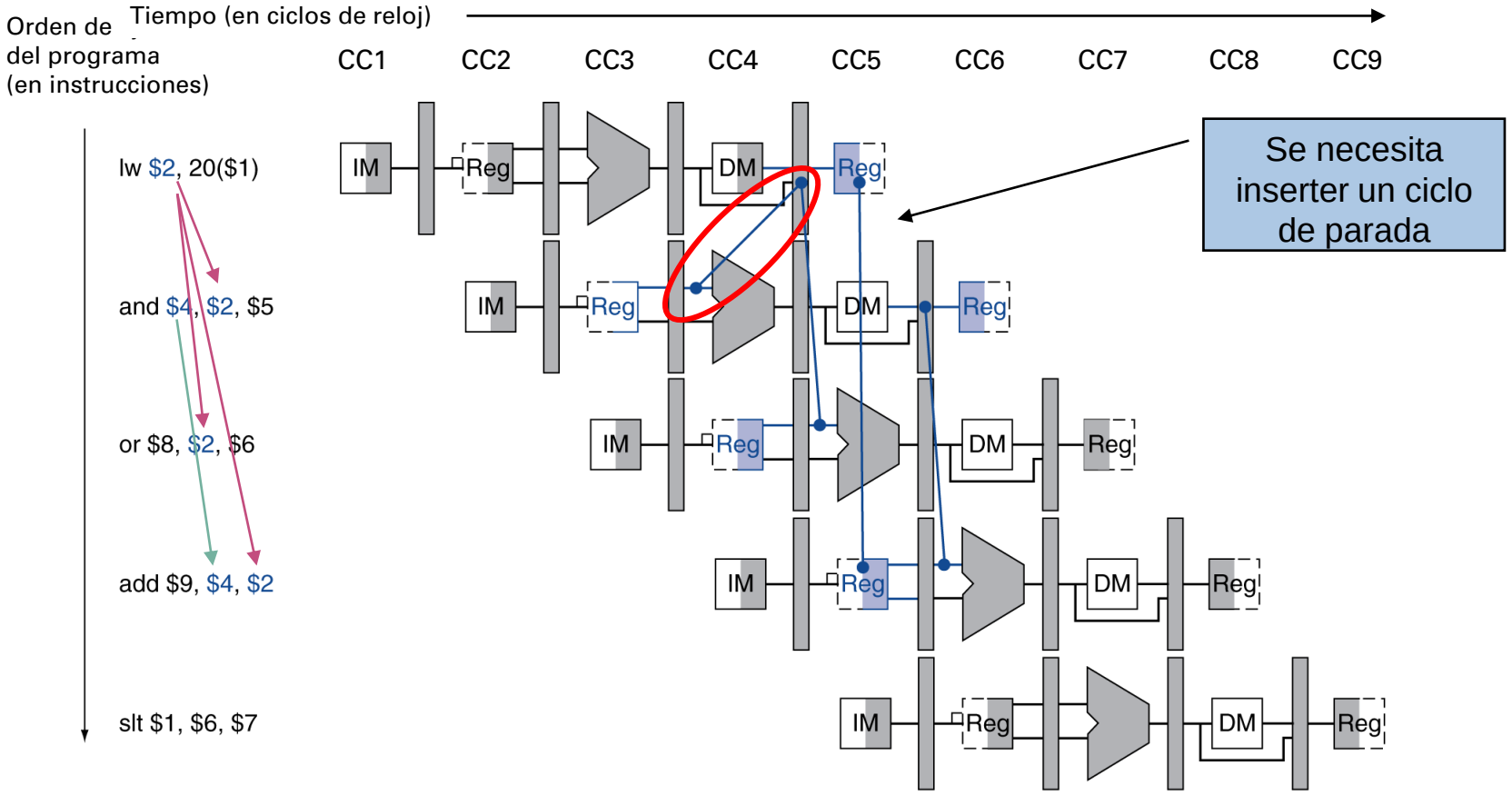
- Requiere de conexiones extra en la ruta de datos.

Ruta de datos con forwarding



Riesgo de datos que requieren detenciones

- *Lw* no tiene el dato hasta final del ciclo 4, la instrucción *and* lo necesita al inicio de ese ciclo
- Podemos adelantar a la ALU desde MEM/WB para *or* ($2) y *add* ($4). La *add ($2)* no tiene problemas ya que recibe el valor a través del banco de registros.



Riesgo de datos que requieren detenciones

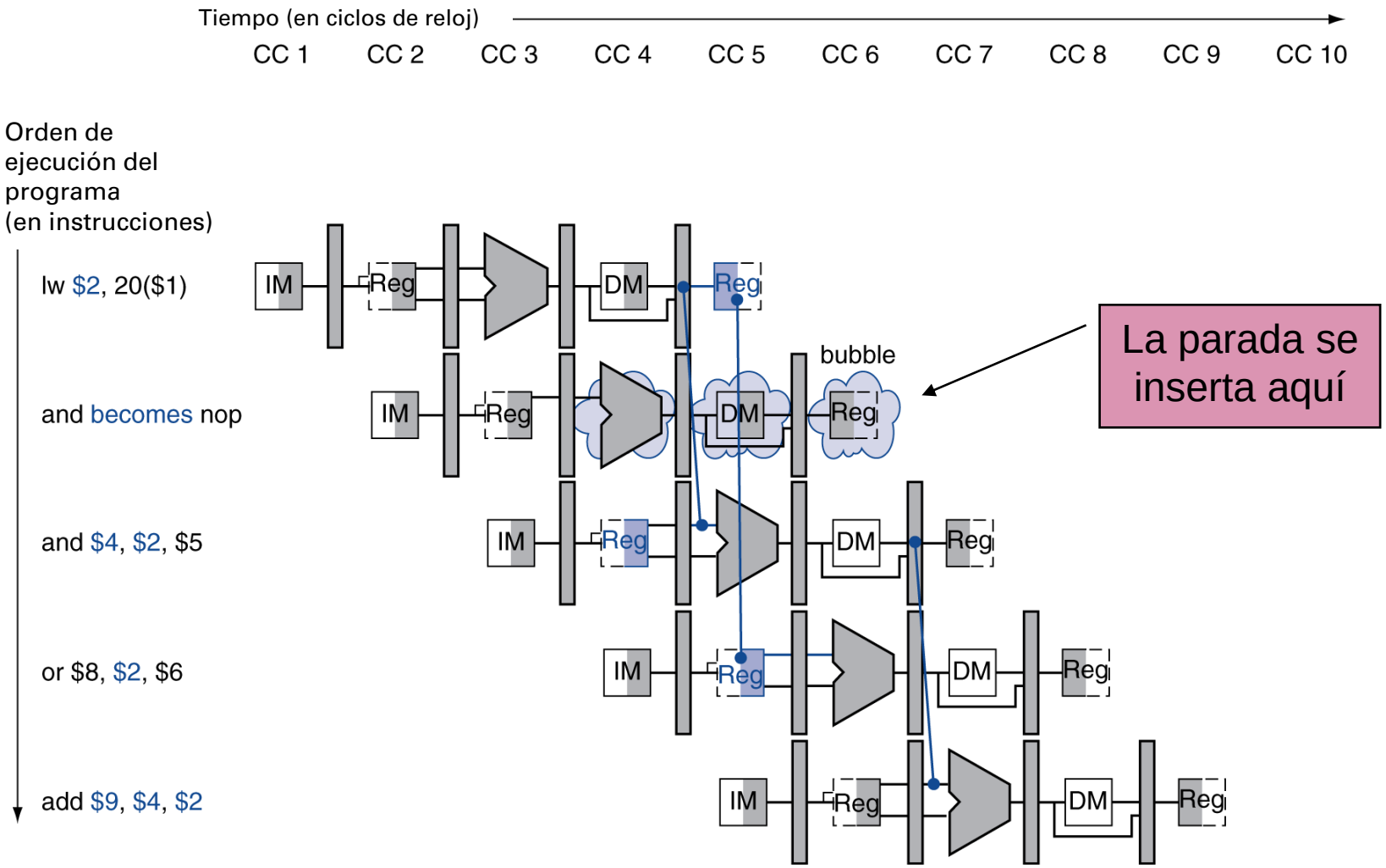
- Ahora el adelantamiento para *and* es correcto a través del registro MEM/WB
- La *or* ($2) no tiene problemas ya que recibe el valor a través del banco de registros.
- El adelantamiento para *add* ($4) continúa siendo a través del registro MEM/WB

Ciclo	1	2	3	4	5	6	7	8	9	10
lw $2, 20($1)	IF	ID	EX	MEM	WB					
And $4, $2, $5		IF	ID	**Parada**	EX	MEM	WB			
or $8, $2, $6			IF	**Parada**	ID	EX	MEM	WB		
Add $9, $4, $2				**Parada**	IF	ID	EX	MEM	WB	
slt $1, $6, $7						IF	ID	EX	MEM	WB

Como para la segmentación

- Forzar los valores de control en el registro ID/EX a 0
  - EX, MEM y WB serán nop (no-operación).
- Prevenir actualizar el PC en el registro IF/ID
  - La instrucción en uso se decodifica de nuevo
  - La siguiente instrucción se busca otra vez
  - 1 ciclo de parada permite que en la etapa MEM se lea el dato para la lw
    - Se puede adelantar a la etapa EX.

Cómo usa la lw la detención de parada



Detenciones y rendimiento

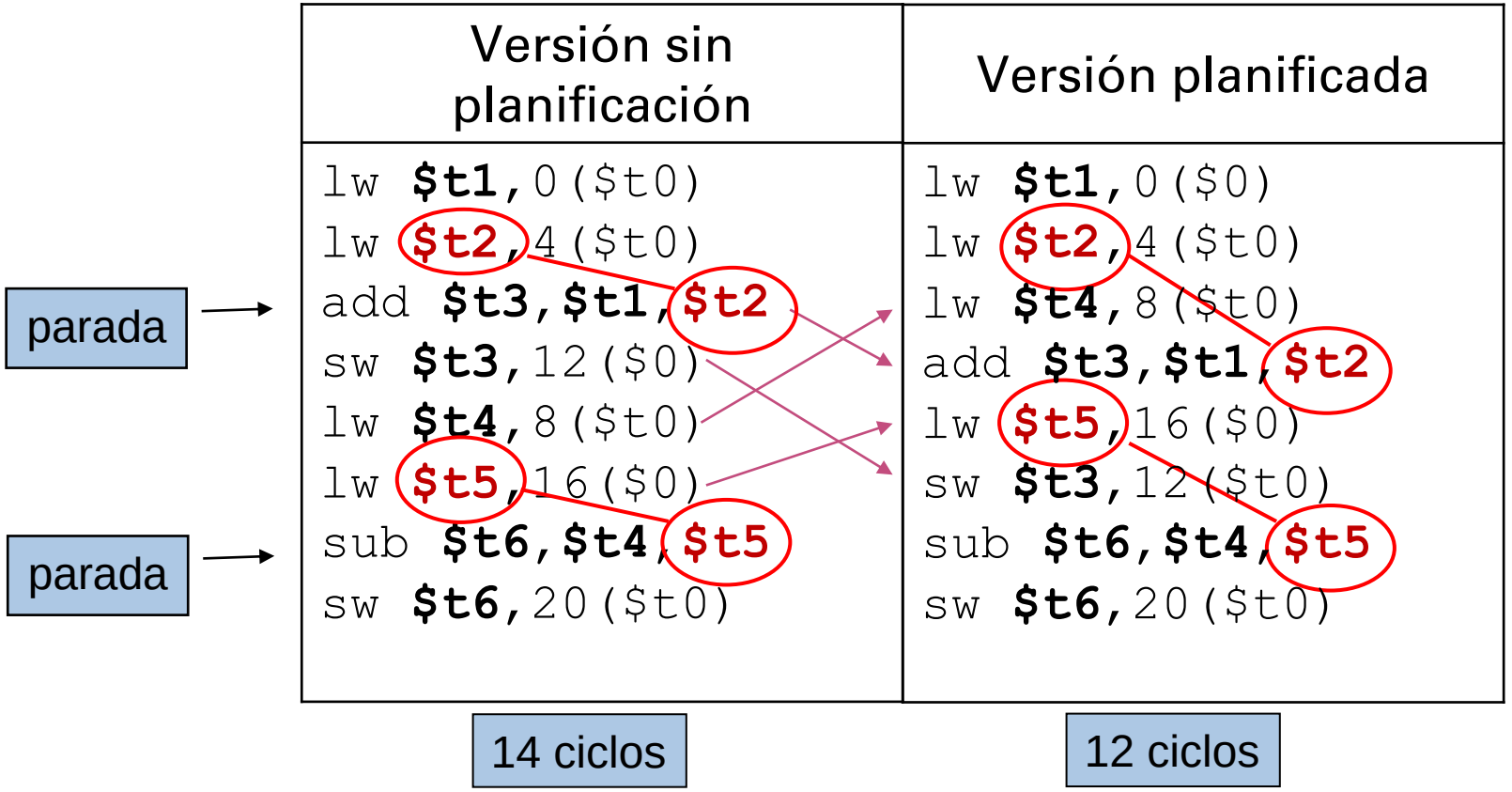
- Las detenciones reducen el rendimiento, pero son necesarias para conseguir resultados correctos
- El compilador puede reordenar el código para evitar riesgos y paradas
- El compilador necesita conocer la estructura de la segmentación

Reordenar el código para evitar paradas

- Generar código que evite detenciones para la siguiente secuencia:

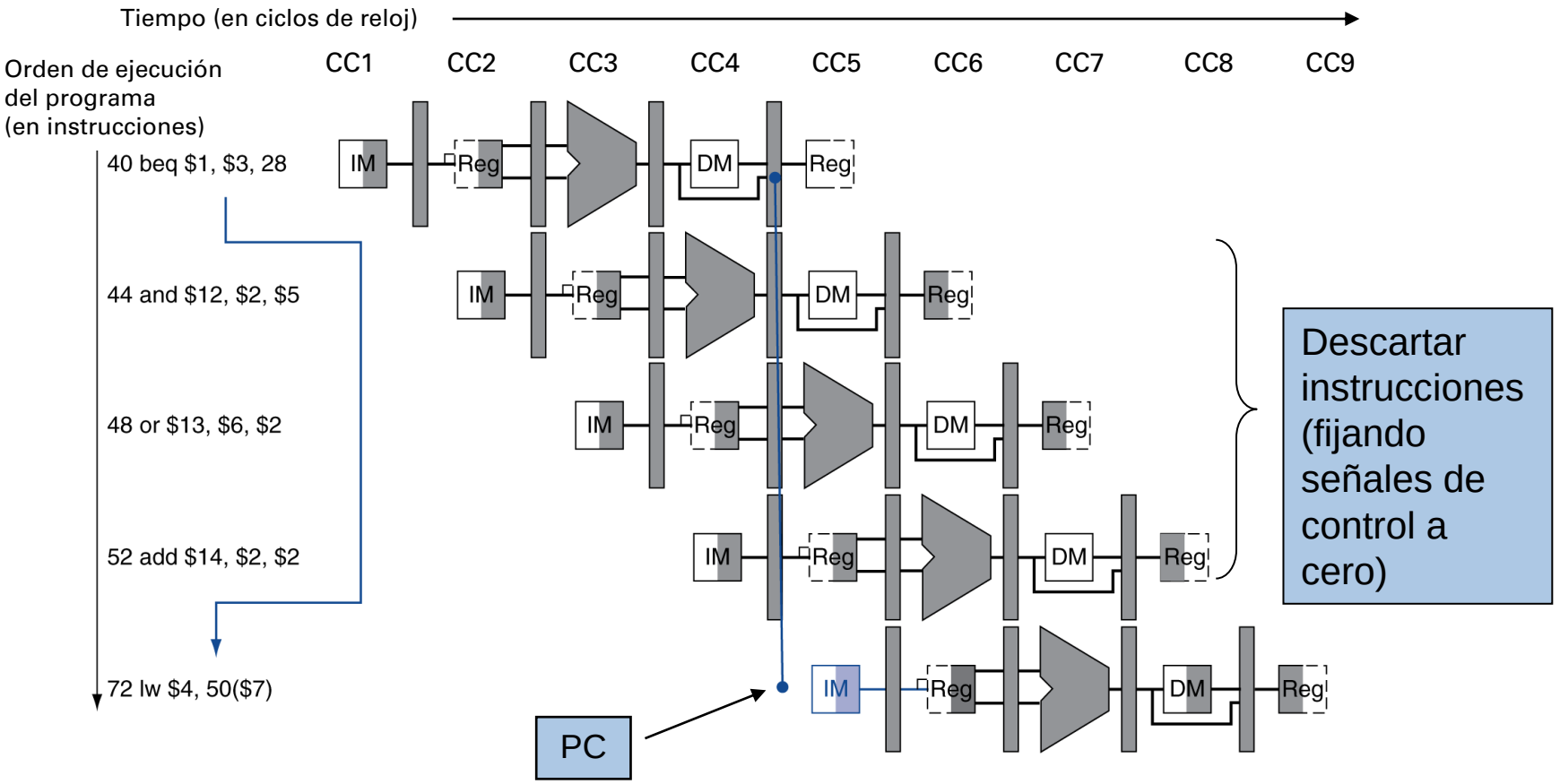
c = a + b

d = e - f



Riesgos de control

- Hasta la etapa MEM no se sabe si el salto es efectivo o no
- Si el salto es efectivo se tendrán que descartar las instrucciones en ejecución hasta el ciclo 5 cuando el PC será correcto.

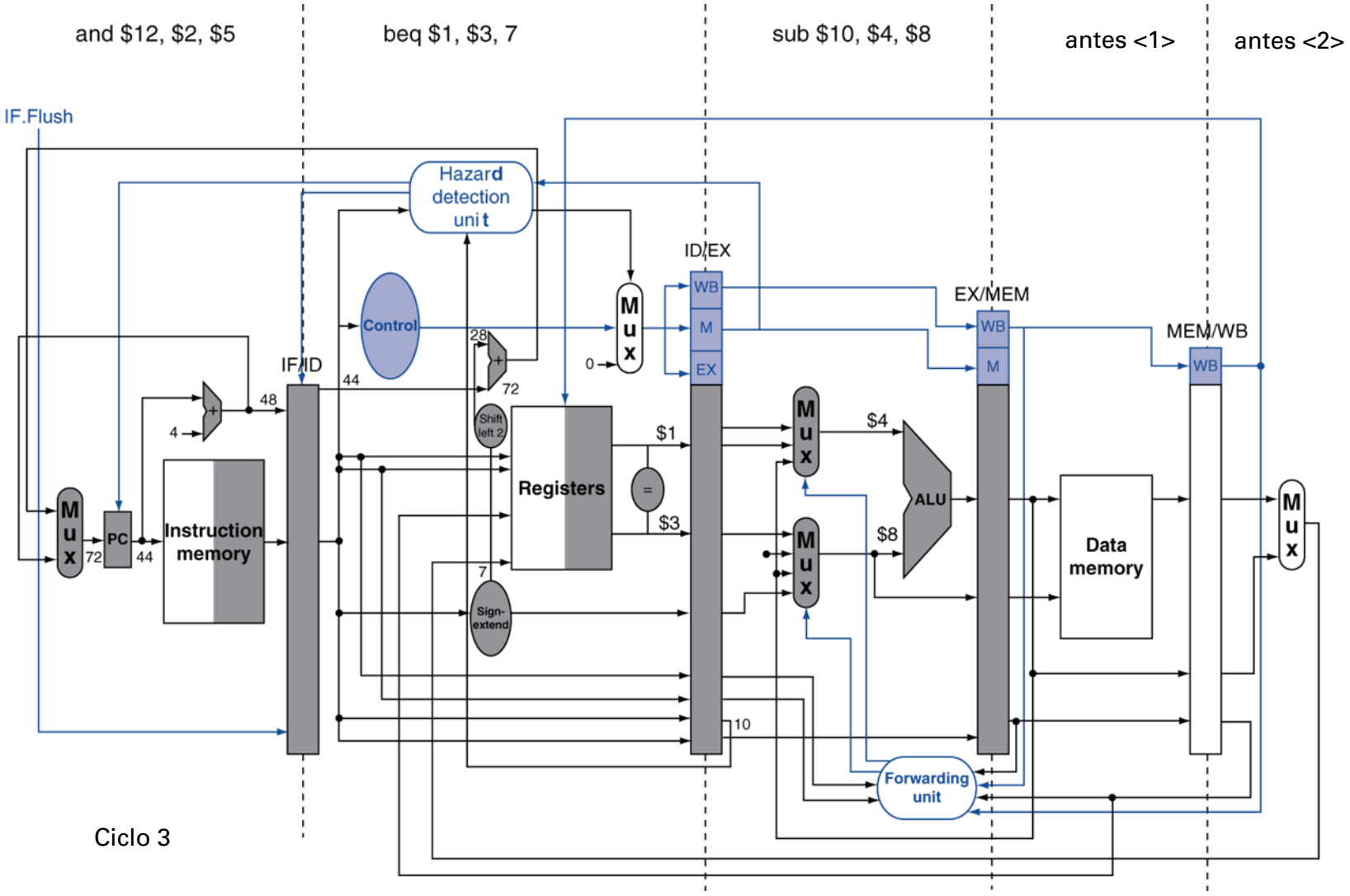


Parada en riesgos de control

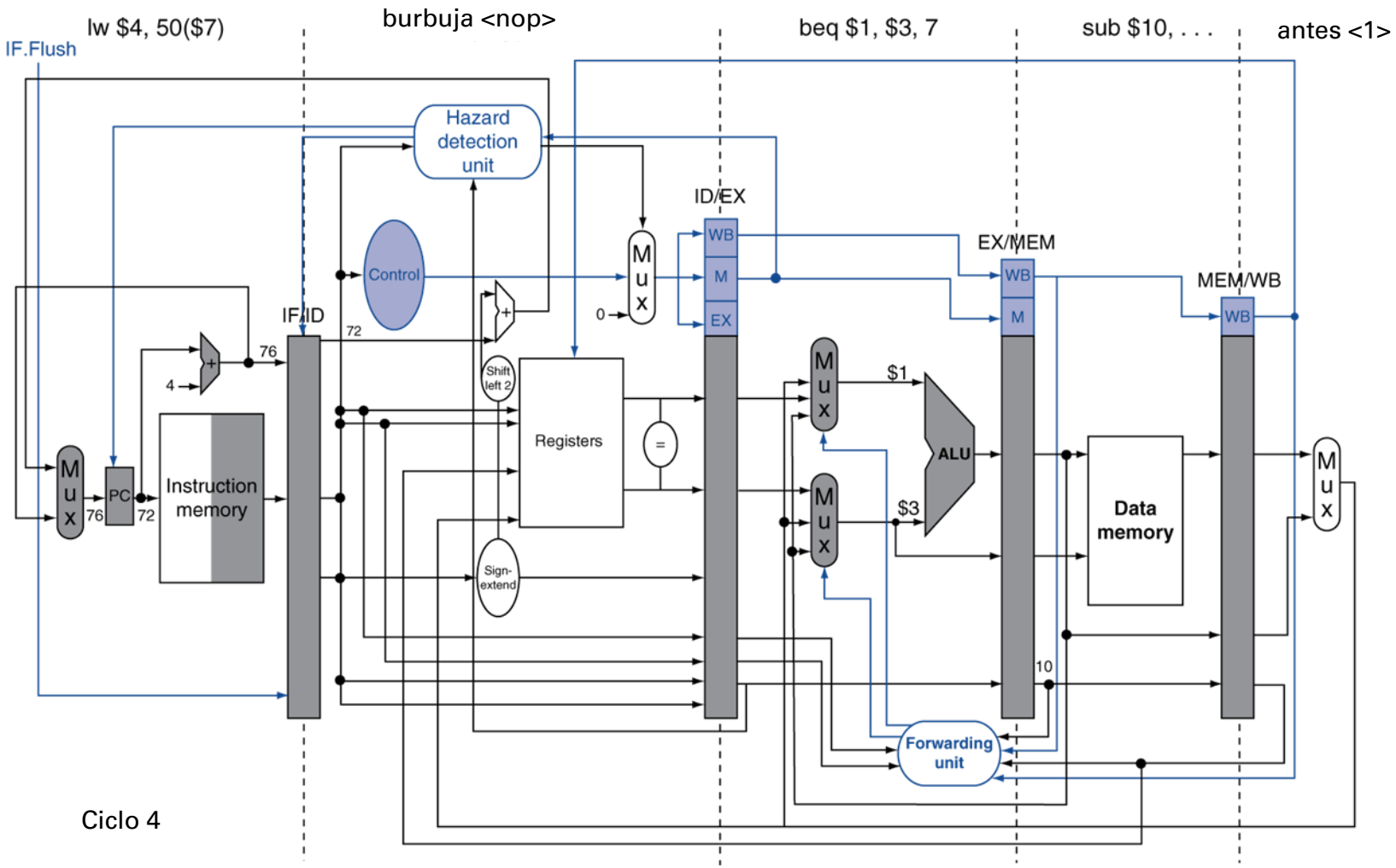
- Será necesario comparar los registros y calcular la dirección de salto tan pronto se pueda
- Se añade hardware para hacerlo en la etapa ID (Sumador para el cálculo dirección de salto y comparador de registros)
- Esperar hasta que se determine el salto antes de buscar la siguiente instrucción

Ciclo	1	2	3	4	5	6	7	8	9	10
Add $4, $5, $6	IF	ID	EX	MEM	WB					
Beq $1, $2, 40		IF	ID	EX	MEM	WB				
or $7, $8, $9			**Parada**	IF	ID	EX	MEM	WB		
or $8, $2, $6					IF	ID	EX	MEM	WB	
slt $1, $6, $7						IF	ID	EX	MEM	WB

Ejemplo: Salto tomado



Ejemplo: Salto tomado



Predicción de saltos

- Reducción de las penalizaciones de saltos en la segmentación: **Predecir el salto como no efectivo**
  - Cuando el salto es no efectivo (se determina en ID), simplemente continuamos
  - Si en ID se determina que el salto es efectivo se busca la instrucción de salto y se deshacen los cambios (poniendo a cero las señales de control). Esto hace que todas las instrucciones que siguen al salto se detengan un ciclo de reloj.
- Predecir el salto como efectivo: no aplicable al MIPS

Predecir el salto como no efectivo

Predicción correcta

Ciclo	1	2	3	4	5	6	7	8
Add $4, $5, $6	IF	ID	EX	MEM	WB			
Beq $1, $2, 40		IF	ID	EX	MEM	WB		
Lw $3, 300($0)			IF	ID	EX	MEM	WB	
…				IF	ID	EX	MEM	WB
….					….			
Or $7,$8,$9								

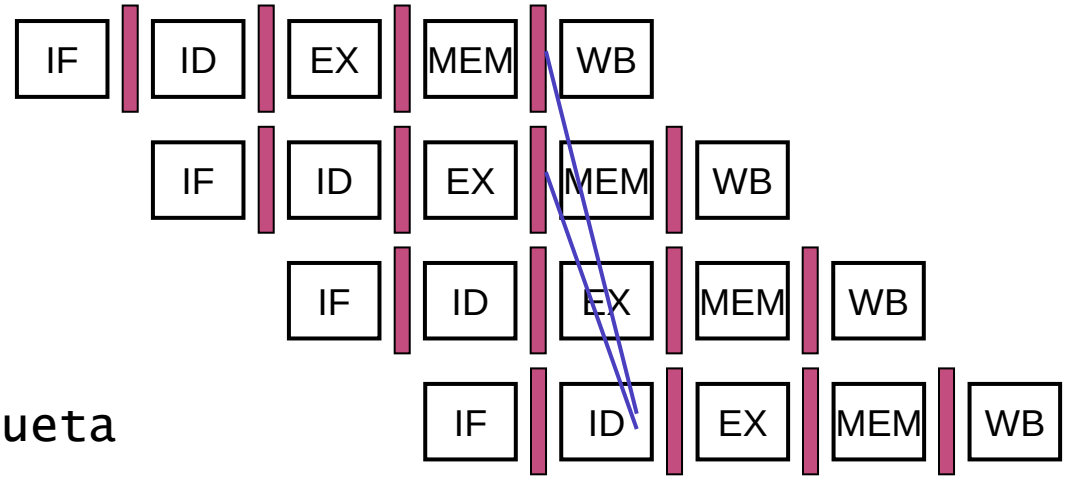
Predicción incorrecta

Ciclo	1	2	3	4	5	6	7	8
Add $4, $5, $6	IF	ID	EX	MEM	WB			
Beq $1, $2, 40		IF	ID	EX	MEM	WB		
Lw $3, 300($0)			IF	burbuja	burbuja	burbuja	burbuja	
…								
Or $7,$8,$9				IF	ID	EX	MEM	WB

Riesgos de datos en saltos

- Si un registro a comparar es destino de la segunda o tercera instrucción ALU anterior
- Se puede resolver con forwarding

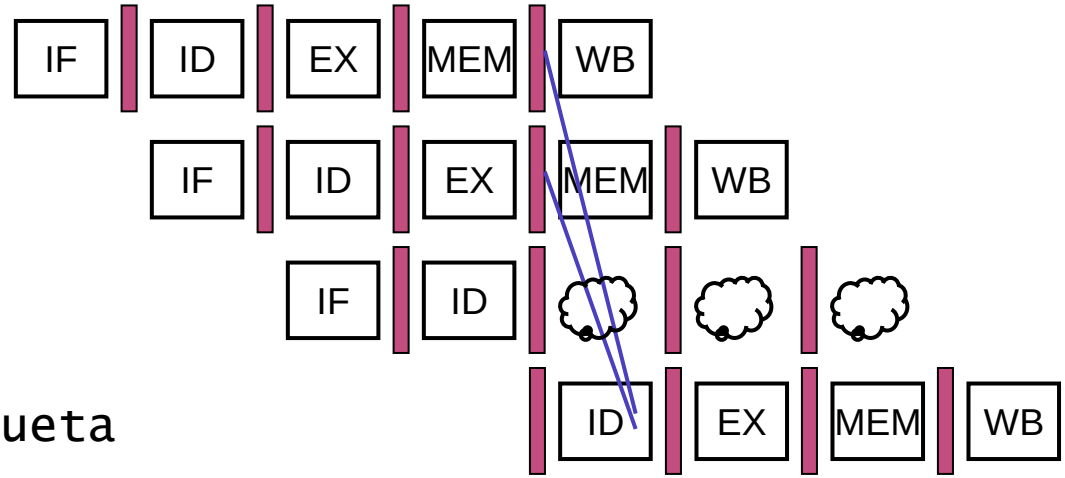
```
add $1, $2, $3
add $4, $5, $6
…
beq $1, $4, etiqueta
```



Riesgos de datos en saltos

- Si un registro a comparar es destino de la instrucción ALU anterior o la segunda instrucción load anterior
- Se necesita un ciclo de parada

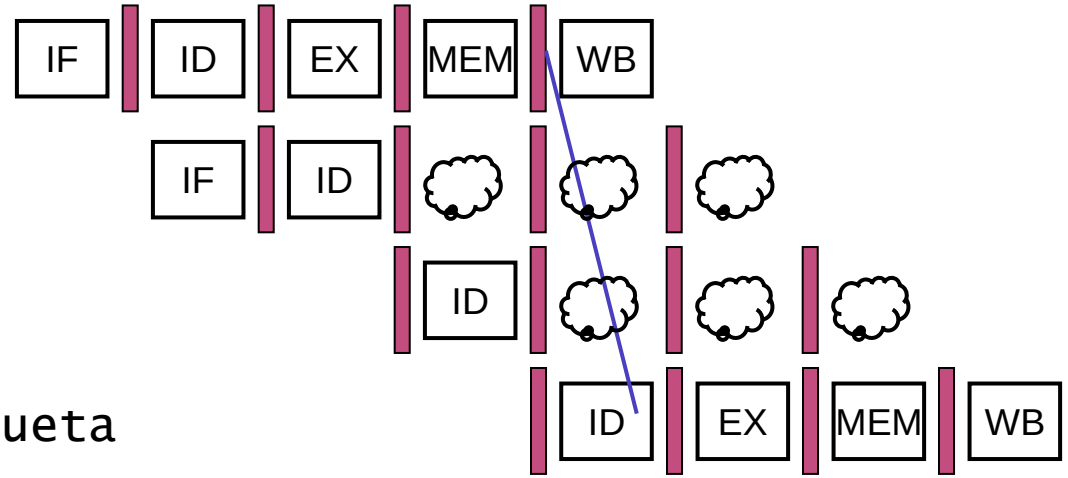
```
lw   $1, 0($3)
add  $4, $5, $6
beq  detenida
beq  $1, $4, etiqueta
```



Riesgos de datos en saltos

- Si un registro a comparar es destino de una instrucción load que la precede
- Se necesita dos ciclos de parada

```
lw   $1, 0($3)
beq  detenida
beq  detenida
beq $1, $0, etiqueta
```

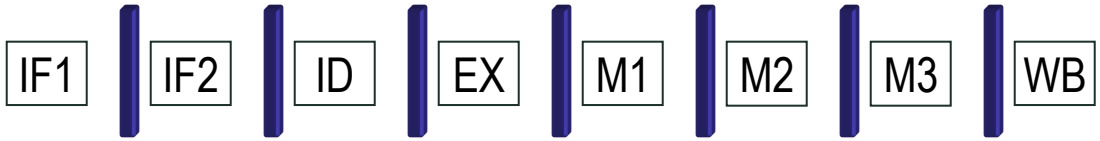


Reducción penalización en saltos

- En segmentaciones profundas y en superescalares, las penalizaciones por saltos son más significativas
- Predicción estática de saltos:
  - Se basa en el comportamiento típico de los saltos (Ejemplo: bucles y declaraciones *if)*
    - Se predicen los saltos hacia atrás como tomados y los saltos hacia adelante como no tomados
- Predicción dinámica de saltos
  - El hardware mide el comportamiento de los saltos (por ejemplo, registrando la historia reciente de cada salto)
  - Se supone que el comportamiento futuro continuará con la misma tendencia
    - Cuando es erróneo, se detiene mientras se vuelve a buscar la instrucción correcta y se actualiza la historia
- Saltos retardados (Ha ido perdiendo vigencia con los años frente a la predicción dinámica)
  - Siempre se ejecuta la instrucción después del salto (ciclo de retardo: delay-slot)
  - Trabajo del compilador: hacer que la instrucción en el delay slot sea válida y útil
  - Si no se encuentra una instrucción poner *nop*

4. Paralelismo a nivel de instrucciones

- La segmentación explota el paralelismo entre instrucciones. Este paralelismo se llama ILP (instruction-level parallelism). Dos formas de aumentar el potencial:
- Supersegmentación: Incrementar la profundidad de la segmentación reduciendo el trabajo por etapa (más instrucciones ejecutándose en paralelo). El rendimiento aumenta al reducirse el ciclo de reloj
- CPI ideal=1
  - Ejemplo: MIPS R4000: 8 etapas
    - IF1 y IF2 hacen la lectura de instrucción
    - M1, M2 y M2 realizan el acceso a la memoria de datos



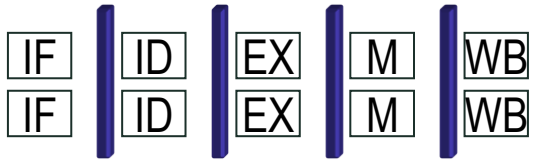
4. Paralelismo a nivel de instrucciones

- Procesadores superescalares: Cada etapa puede procesar más instrucciones.

  Se replican las etapas (varios pipelines) :

  - Se inician varias instrucciones en cada ciclo de reloj.
  - CPI<1, se usa IPC (instrucciones por ciclo)
  - Por ejemplo, con 4 instrucciones a 4GHz
    - 16 BIPS (16 billones de instrucciones por segundo), CPI ideal=0,25, IPC ideal= 4
    - En una segmentación de 5 etapas podría haber 20 instrucciones en ejecución en un instante dado
    - En la práctica, las dependencias reducen estos valores.
  - Con dos instrucciones (CPI ideal = 0.5; IPC ideal = 2)

Instr.	1	2	3	4	5	6	7
i1	IF	ID	EX	M	WB		
i2	IF	ID	EX	M	WB		
i3		IF	ID	EX	M	WB	
i4		IF	ID	EX	M	WB	
i5			IF	ID	EX	M	WB
i6			IF	ID	EX	M	WB



Procesadores superescalares

- Planificación estática:
  - El compilador agrupa instrucciones para ser lanzadas juntas
  - Las empaqueta en lo que se denomina “issues slot”
  - El compilador detecta y evita los riesgos
- Planificación dinámica:
  - La CPU examina la cadena de instrucciones y elige que instrucciones lanzar en cada ciclo de reloj
  - El compilador puede ayudar reordenando las instrucciones
  - La CPU resuelve los riesgos utilizando técnicas avanzadas en tiempo de ejecución

5. Conclusiones

- El diseño del repertorio de instrucciones influye en el diseño de la ruta de datos y control
- El diseño de la ruta de datos y el control influye en el diseño del repertorio de instrucciones
- La segmentación mejora la productividad utilizando paralelismo
  - Más instrucciones completadas por segundo
  - La latencia de cada instrucción no se reduce
- Riesgos de la segmentación: estructurales, de datos y control
- Procesadores superescalares y planificación dinámica (ILP)
  - Las dependencias limitan el paralelismo alcanzable
  - La complejidad conduce al “power wall” (se refiere a las dificultades de aumentar el rendimiento del procesador debido a las limitaciones en el suministro y disipación de potencia asequibles)