Tema 4:
Segmentación *-pipelining-* 1ª parte

Arquitectura de Computadores

Grado en Ingeniería Informática

Mercedes Rodríguez García

Índice

1. Segmentación *-pipelining-*
2. Consideraciones en el diseño del repertorio de instrucciones
3. Camino de datos de un procesador segmentado
4. Representaciones gráficas de la segmentación
5. Riesgos del pipeline
   - 5.1. Riesgos estructurales
   - 5.2. Riesgos de datos
     - 5.2.1. Métodos para resolver los riesgos de datos
     - 5.2.2. Tipos de riesgos de datos
   - 5.3. Riesgos de control
     - 5.3.1. Métodos para resolver los riesgos de control
6. ILP avanzado
   - 6.1. Estrategias para incrementar ILP
   - 6.2. Ejecución múltiple con planificación estática
   - 6.3. Ejecución múltiple con planificación dinámica
7. Casos reales

Bibliografía

Estructura y diseño de computadores: Capítulo 4.
Patterson y Hennessy. Editorial Reverte, 2011.

1. Segmentación *-pipelining-*

1. Segmentación *-pipelining-*
2. Consideraciones en el diseño del repertorio de instrucciones
3. Camino de datos de un procesador segmentado
4. Representaciones gráficas de la segmentación
5. Riesgos del pipeline
   - 5.1. Riesgos estructurales
   - 5.2. Riesgos de datos
     - 5.2.1. Métodos para resolver los riesgos de datos
     - 5.2.2. Tipos de riesgos de datos
   - 5.3. Riesgos de control
     - 5.3.1. Métodos para resolver los riesgos de control
6. ILP avanzado
   - 6.1. Estrategias para incrementar ILP
   - 6.2. Ejecución múltiple con planificación estática
   - 6.3. Ejecución múltiple con planificación dinámica
7. Casos reales

1. Segmentación *-pipelining-*

1 **¿Qué es la segmentación?**

Técnica utilizada en el diseño de procesadores para implementar ILP.

IDEA PRINCIPAL: que en un solo camino de datos se estén ejecutando varias instrucciones a la vez, cada una en una fase de ejecución diferente.

Hoy en día la mayoría de los procesadores implementan esta técnica.

2 **Sobre las etapas de segmentación …**

Cada fase de ejecución constituirá una etapa de segmentación.

El pipeline que estudiaremos tendrá 5 etapas de segmentación.

1. Segmentación *-pipelining-*

3 **¿Cuánto dura cada etapa de segmentación?**

Cada etapa de segmentación dura un ciclo de reloj. La duración de ese ciclo debe ser suficientemente largo como para dar cabida a la operación más lenta.

4 **Ventajas**

- Mejora la productividad.
- Reduce el tiempo total en el que se ejecuta un CONJUNTO de instrucciones.

! La segmentación **no** reduce el tiempo en el que se ejecuta UNA instrucción.

1. Segmentación *-pipelining-*

Ejemplo

Sea un procesador en el que los accesos a memoria duran 200 ps, las operaciones de la ALU 200 ps y los accesos al banco de registros 100 ps. Picosegundo: 1 ps = 10^{-12} s. **-[PATT11]-**

¿Cuál debería ser la duración de un ciclo de reloj en un procesador segmentado?

2. Consideraciones en el repertorio de instrucciones

1. Segmentación *-pipelining-*
2. Consideraciones en el diseño del repertorio de instrucciones
3. Camino de datos de un procesador segmentado
4. Representaciones gráficas de la segmentación
5. Riesgos del pipeline
   - 5.1. Riesgos estructurales
   - 5.2. Riesgos de datos
     - 5.2.1. Métodos para resolver los riesgos de datos
     - 5.2.2. Tipos de riesgos de datos
   - 5.3. Riesgos de control
     - 5.3.1. Métodos para resolver los riesgos de control
6. ILP avanzado
   - 6.1. Estrategias para incrementar ILP
   - 6.2. Ejecución múltiple con planificación estática
   - 6.3. Ejecución múltiple con planificación dinámica
7. Casos reales

2. Consideraciones en el diseño del repertorio de instrucciones

Arquitectura MIPS: "una arquitectura expresamente diseñada para la segmentación"

1. Todas las instrucciones tienen la **misma longitud**.

→ Facilita la búsqueda de instrucciones.

2. **Pocos formatos** de instrucciones y prácticamente todos tienen los campos de registros fuentes en la misma posición de la instrucción.

→ Menos etapas en el pipeline: la segunda etapa puede empezar a leer los operandos en los registros al mismo tiempo que decodifica la instrucción.

¿Qué ocurriría si no fuese así? Tendríamos más etapas, por ejemplo:
1. IF, **2.** ID, **3.** lectura registros, **4.** EX, **5.** MEM, **6.** WB

3. Modelo de ejecución **registro-registro**.

→ Menos etapas en el pipeline: los operandos ya están en los registros no hay que ir a buscarlos a memoria.

¿Qué ocurriría si no fuese así? Tendríamos más etapas, por ejemplo:
1. IF, **2.** ID, **3.** Cálculo dirección, **4.** MEM, **5.** EX, **6.** WB

2. Consideraciones en el diseño del repertorio de instrucciones

Arquitectura x86: "una arquitectura que dificulta la segmentación"

Dificultad: la longitud de las instrucciones varía de 1 byte a 17 bytes.

Dificultad: modelo de ejecución registro-memoria.

Solución: Las procesadores recientes convierten las instrucciones x86 en instrucciones simples de tipo RISC (Intel las llama microoperaciones y AMD las llama operaciones RISC o Rops).

Investiga las siguientes arquitecturas:

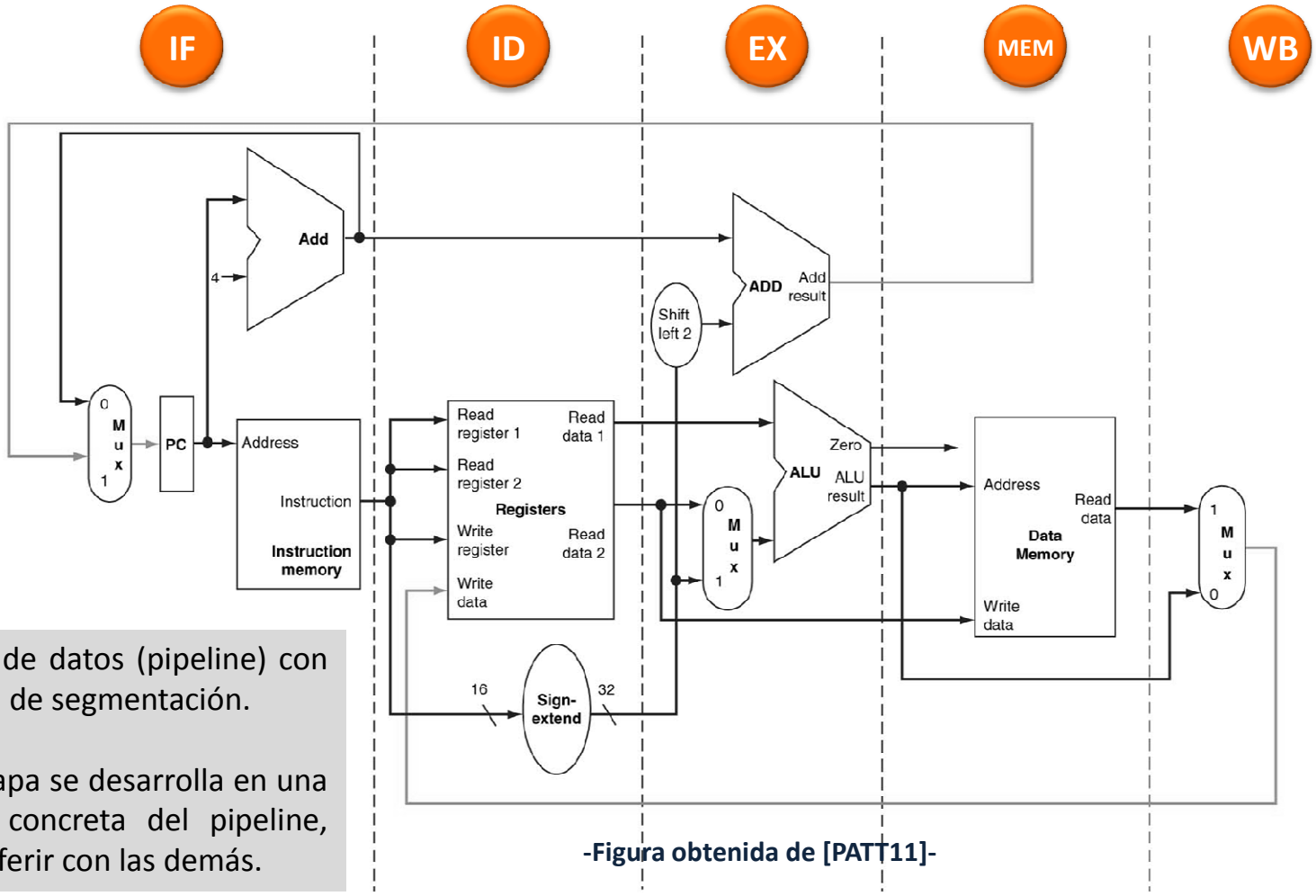
x86-32 ≡ IA32
x86-64
IA64

http://es.wikipedia.org/wiki/IA-32
http://es.wikipedia.org/wiki/X86-64
http://es.wikipedia.org/wiki/Itanium

3. Camino de datos de un procesador segmentado

1. Segmentación *-pipelining-*
2. Consideraciones en el diseño del repertorio de instrucciones
3. Camino de datos de un procesador segmentado
4. Representaciones gráficas de la segmentación
5. Riesgos del pipeline
   - 5.1. Riesgos estructurales
   - 5.2. Riesgos de datos
     - 5.2.1. Métodos para resolver los riesgos de datos
     - 5.2.2. Tipos de riesgos de datos
   - 5.3. Riesgos de control
     - 5.3.1. Métodos para resolver los riesgos de control
6. ILP avanzado
   - 6.1. Estrategias para incrementar ILP
   - 6.2. Ejecución múltiple con planificación estática
   - 6.3. Ejecución múltiple con planificación dinámica
7. Casos reales

3. Camino de datos de un procesador segmentado



Camino de datos (pipeline) con 5 etapas de segmentación.

Cada etapa se desarrolla en una sección concreta del pipeline, sin interferir con las demás.

-Figura obtenida de [PATT11]-

3. Camino de datos de un procesador segmentado

1.- ¿Realmente no interfieren las etapas?

2.- ¿Qué ocurre cuando en el pipeline hay una instrucción en la etapa WB y otra instrucción en la etapa ID?, es decir, ¿Qué ocurre cuando dos instrucciones tienen que acceder al banco de registros en el mismo ciclo de reloj?

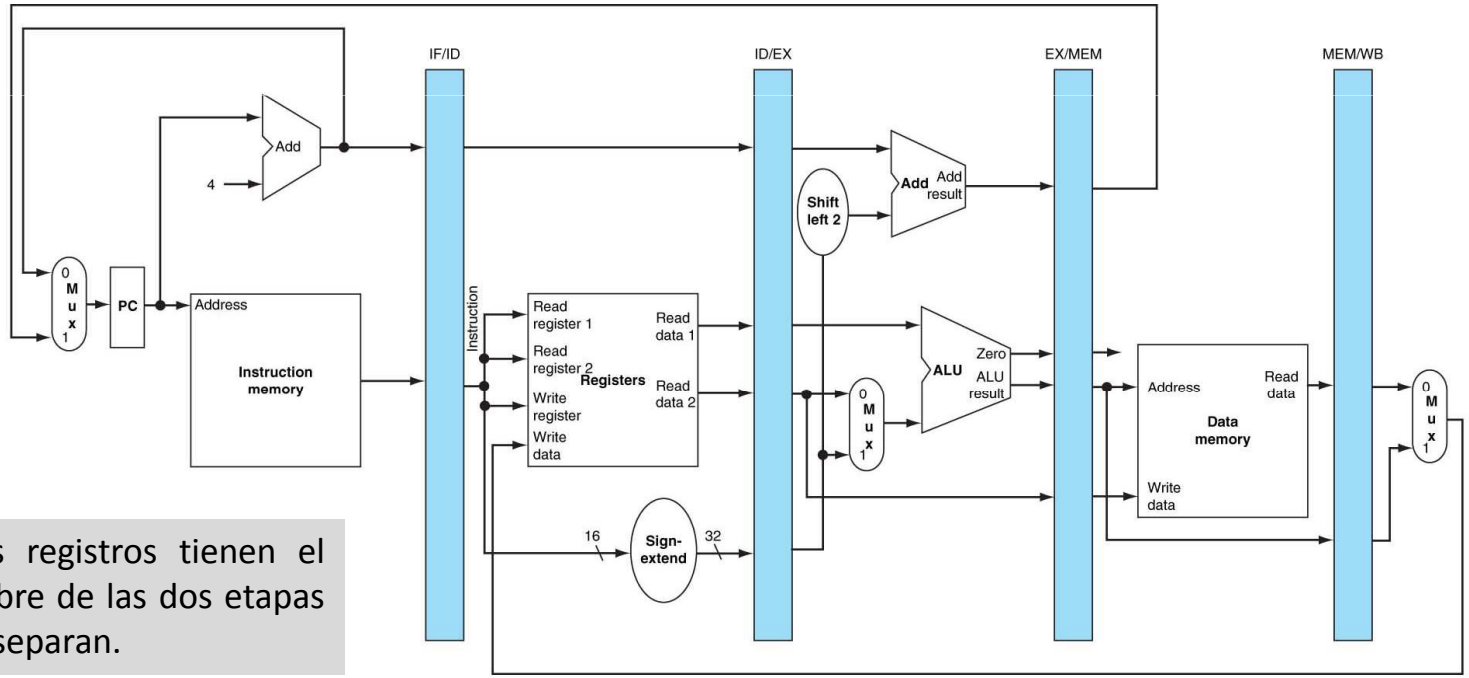
3.- ¿Qué sucedería si la caché L1 fuese una memoria unificada?

3. Camino de datos de un procesador segmentado

SITUACIÓN: Cuando una instrucción X finaliza una etapa, deja de usar esa sección del pipeline y pasa a la siguiente. La sección que ha dejado libre es ocupada ahora por otra instrucción.

PROBLEMA: Cuando una instrucción X pasa a otra etapa, **¿qué ocurre con su contexto de ejecución?**

SOLUCIÓN: La información (contexto) que se quiera conservar después de ejecutar una etapa hay que transferirla a la siguiente mediante los **REGISTROS DE SEGMENTACIÓN**. En caso contrario, la información se perderá por sobreescrituras que realicen otras instrucciones.



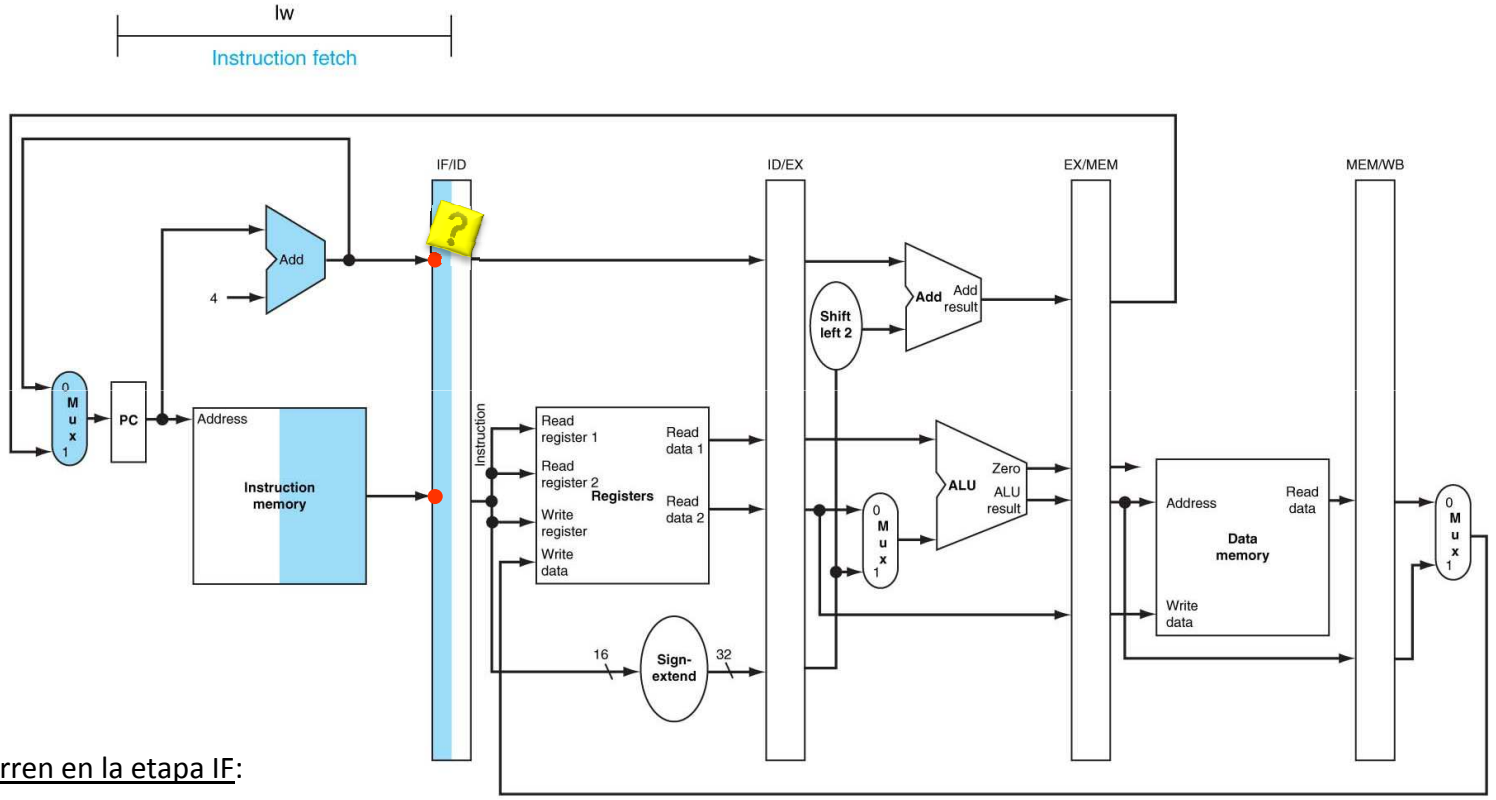
Estos registros tienen el nombre de las dos etapas que separan.

-Figura obtenida de [PATT11]-

3. Camino de datos de un procesador segmentado

Ejemplo

Mostrar la actividad del pipeline a medida que la instrucción **LW $t2, 8($t1)** avanza por las cinco etapas.



-Figura obtenida de [PATT11]-

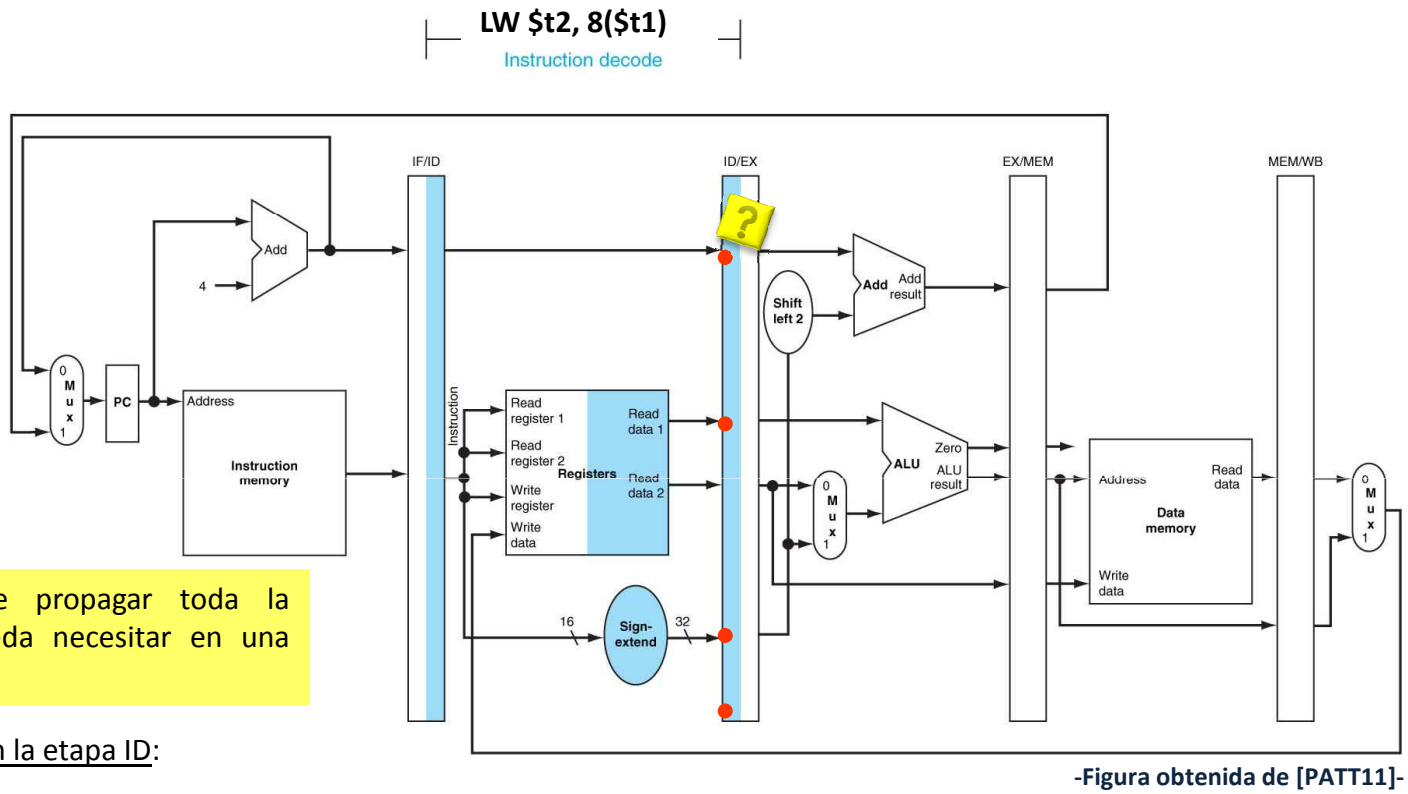
- Acciones que ocurren en la etapa IF:

1. Se lee la instrucción de memoria y se almacena en el registro IF/ID.
2. Se incrementa el PC y se almacena en el registro IF/ID.

- Datos a conservar en el registro IF/ID: la instrucción y el valor del PC.

3. Camino de datos de un procesador segmentado

Ejemplo



RECORDEMOS: hay que propagar toda la información que se pueda necesitar en una etapa posterior.

-Figura obtenida de [PATT11]-

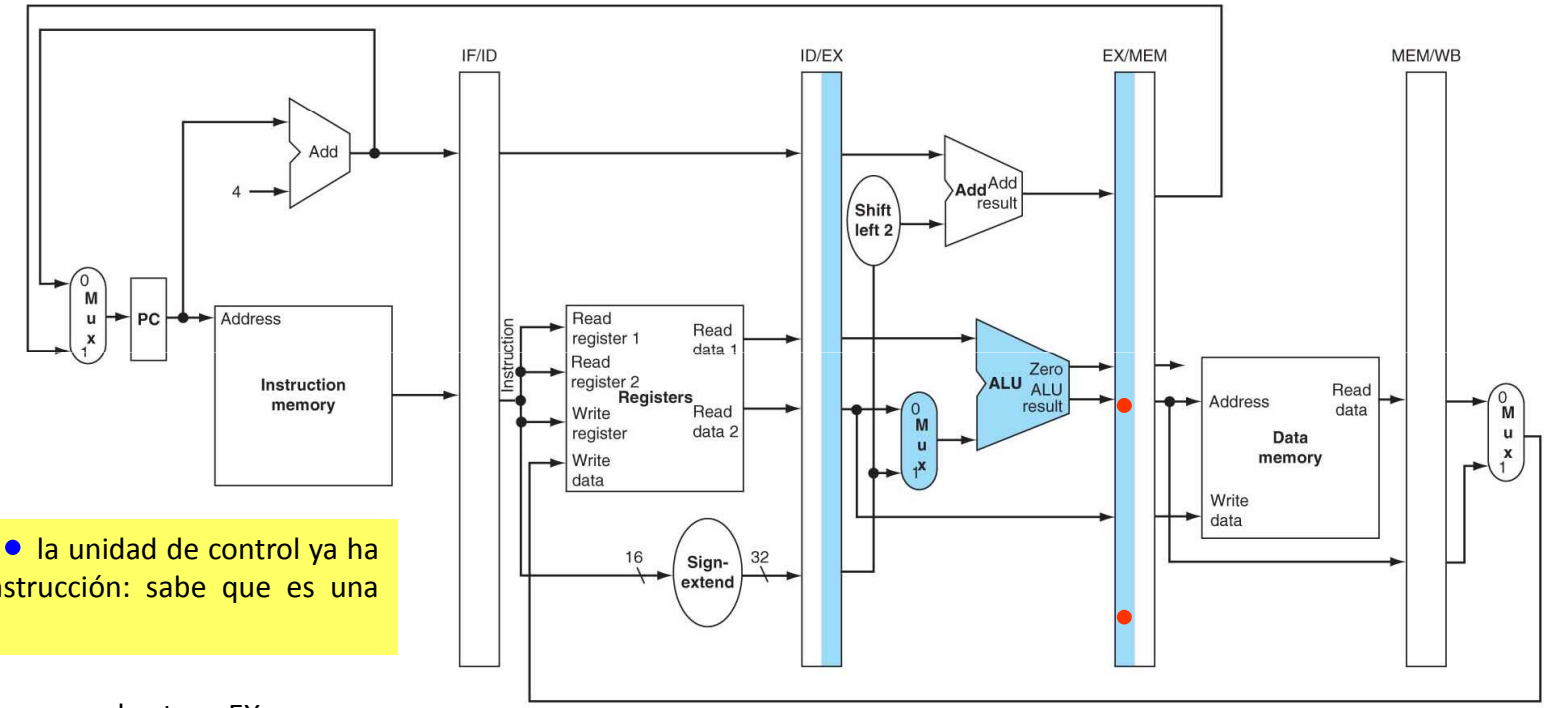
- Acciones que ocurren en la etapa ID:

1. Se decodifica la instrucción.
2. Se lee de IF/ID el número de identificación del registro $t1, se obtiene el operando y se almacena en ID/EX.
3. Se lee de IF/ID el offset, se extiende y se almacena en ID/EX.
4. Se lee de IF/ID el número de identificación del registro destino $t2 y se almacena en ID/EX.
5. Se lee de IF/ID el valor del PC y se almacena en ID/EX.

- Datos a conservar en el registro ID/EX: el operando, el offset, el identificador del registro destino y el valor del PC.

3. Camino de datos de un procesador segmentado

Ejemplo



En este momento ● la unidad de control ya ha decodificado la instrucción: sabe que es una instrucción LW.

-Figura obtenida de [PATT11]-

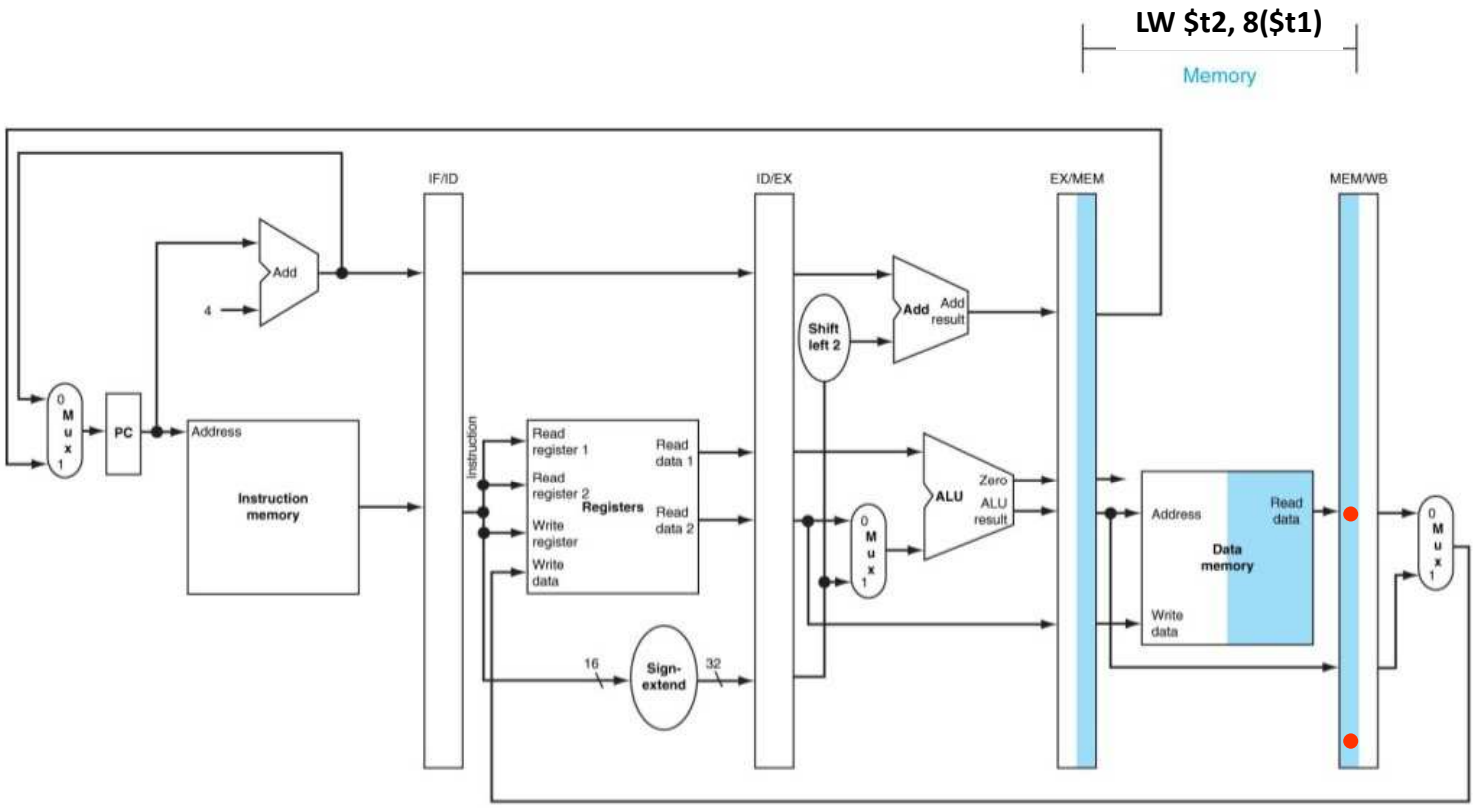
- Acciones que ocurren en la etapa EX:

1. Se lee de ID/EX el operando y el offset, se suman y el resultado se almacena en EX/MEM.
2. Se lee de ID/EX el número de identificación del registro destino $t2 y se almacena en EX/MEM.

- Datos a conservar en el registro EX/MEM: el resultado de la ALU (dirección de memoria) y el identificador del registro destino. ¿Por qué ya no hay que conservar el valor del PC?

3. Camino de datos de un procesador segmentado

Ejemplo



-Figura obtenida de [PATT11]-

- Acciones que ocurren en la etapa MEM:

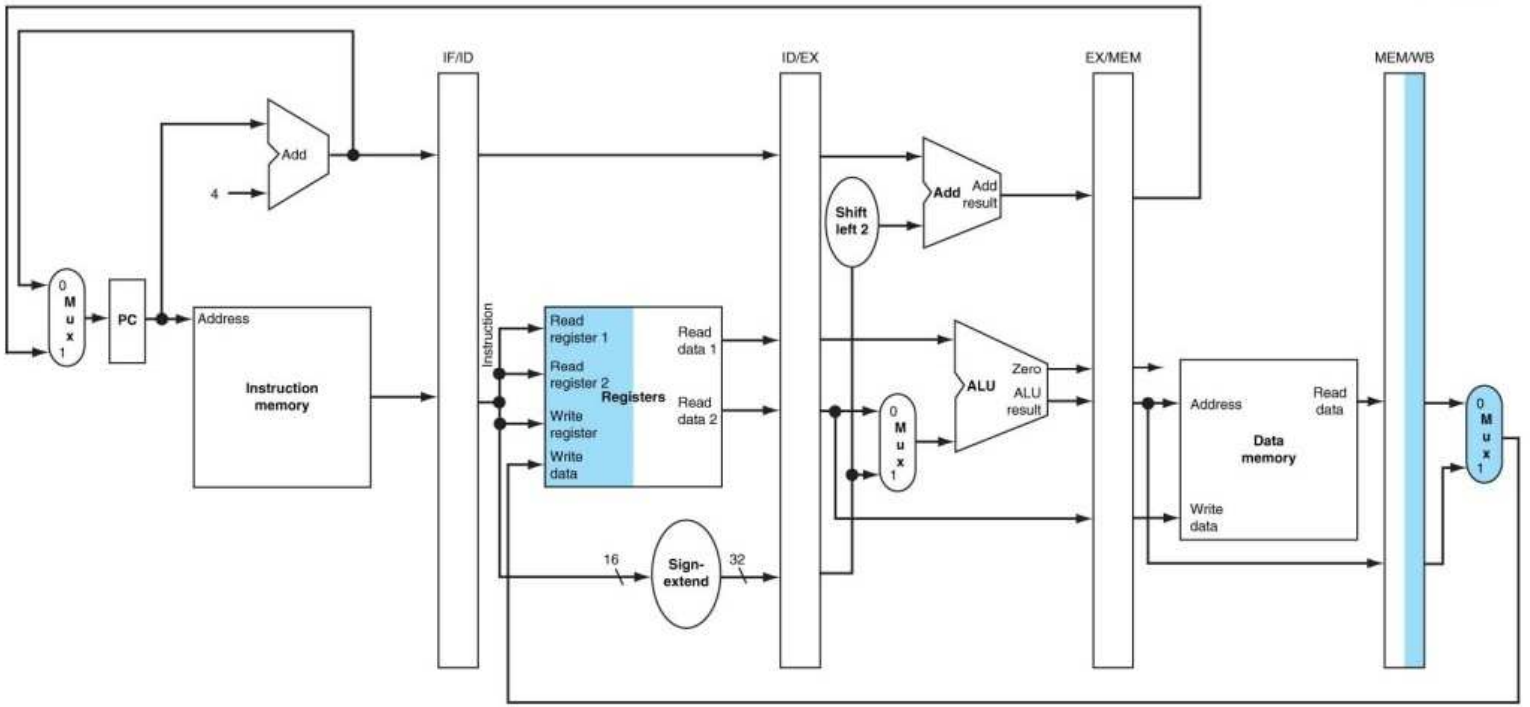
1. Se lee de EX/MEM la dirección calculada en la etapa anterior, se busca el dato en memoria y se almacena en MEM/WB.
2. Se lee de EX/MEM el número de identificación del registro destino $t2 y se almacena en MEM/WB.

- Datos a conservar en el registro MEM/WB: el dato obtenido de memoria y el identificador del registro destino.

3. Camino de datos de un procesador segmentado

Ejemplo

LW $t2, 8($t1)



-Figura obtenida de [PATT11]-

- Acciones que ocurren en la etapa WB:

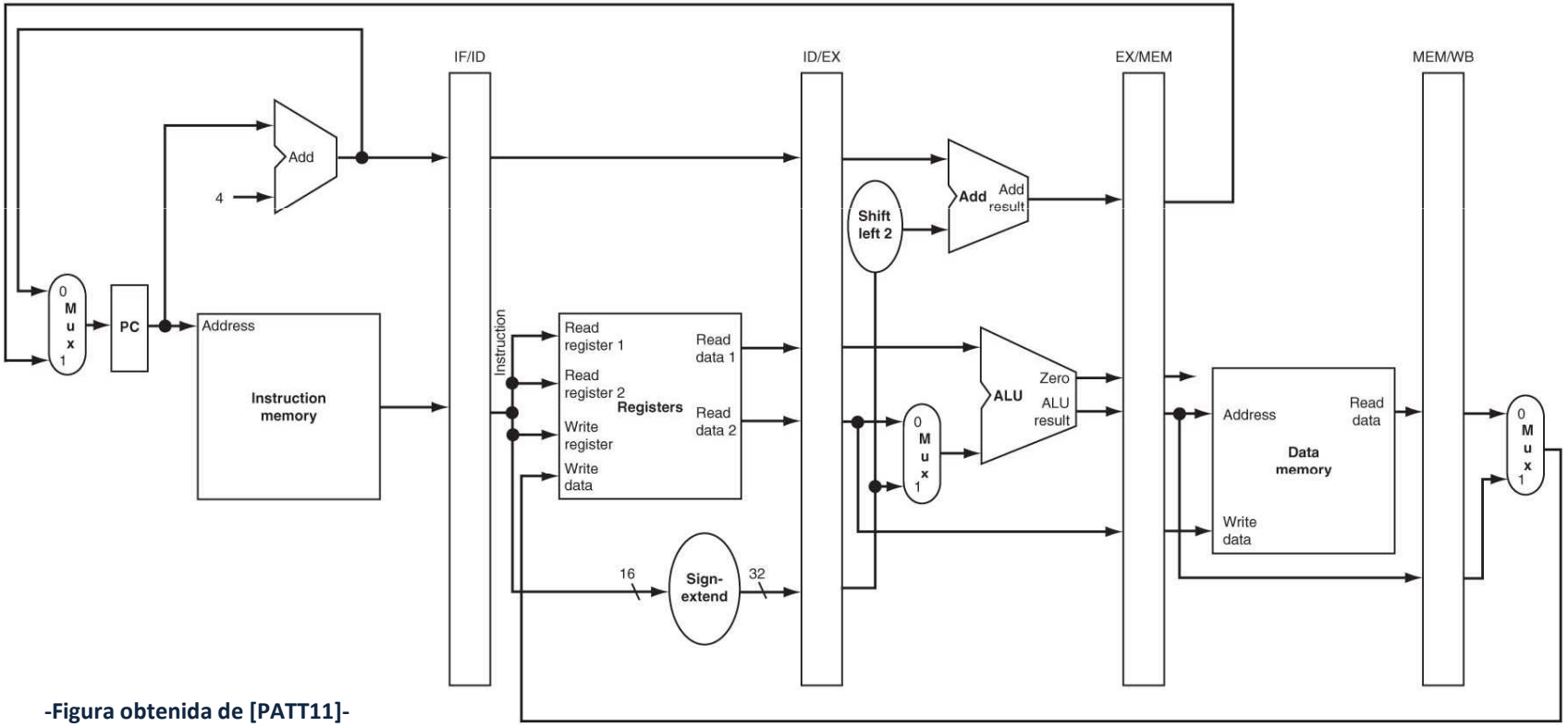
1. Se lee de MEM/WB el dato y el identificador del registro destino. A continuación, se almacena el dato en ese registro.

FIN

3. Camino de datos de un procesador segmentado

Problema 1

Mostrar la actividad del pipeline a medida que la instrucción **SW $t2, 4($t1)** avanza por las cinco etapas.

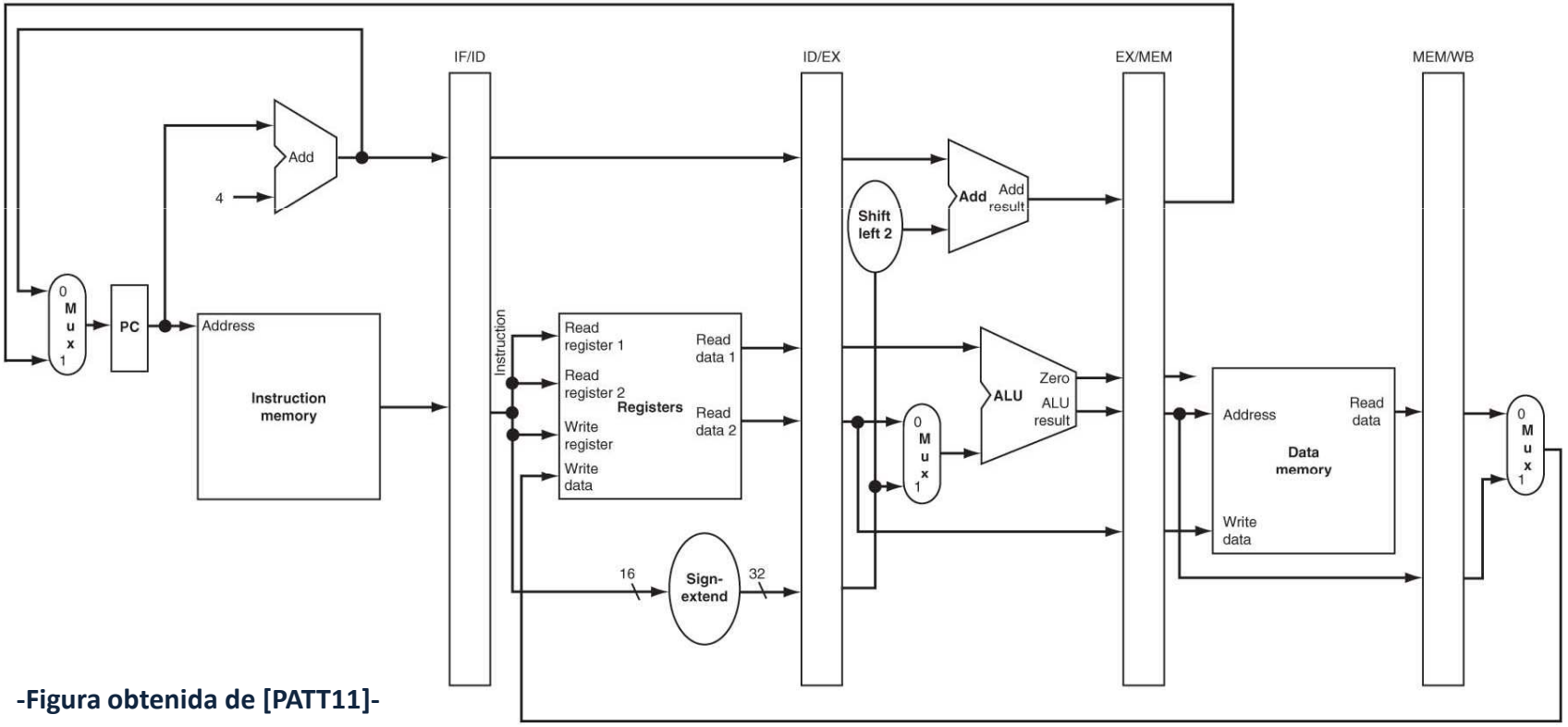


-Figura obtenida de [PATT11]-

3. Camino de datos de un procesador segmentado

Problema 2

Mostrar la actividad del pipeline a medida que la instrucción **ADD $t1, $t2, $t3** avanza por las cinco etapas.



-Figura obtenida de [PATT11]-