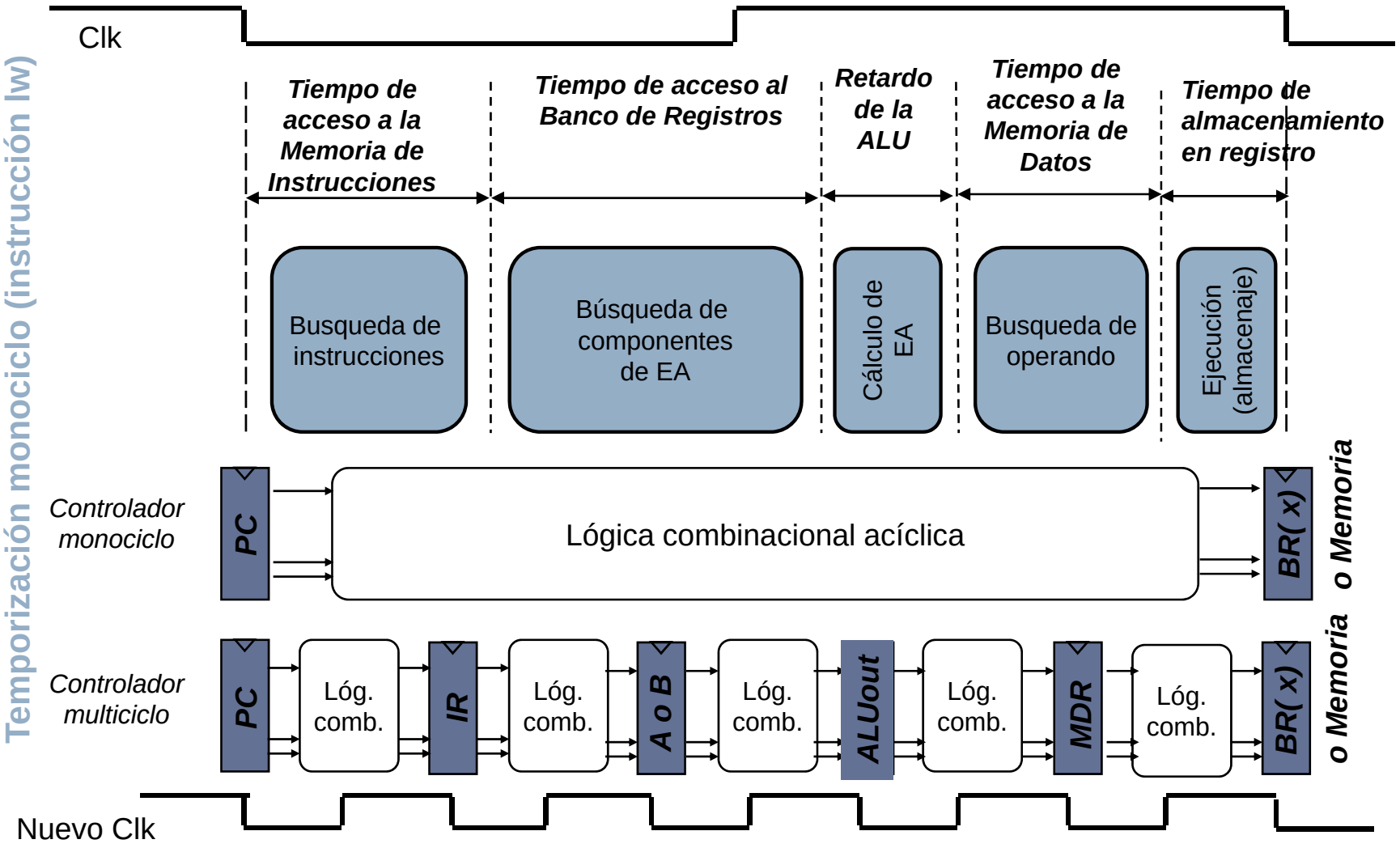
Diseño de la Ruta de Datos y la Unidad de Control

Ruta de datos multiciclo

Contenidos

- Introducción
  - Importancia del diseño del procesador. Metodología de diseño de un procesador. Arquitectura MIPS: formato de la instrucción máquina y repertorio de instrucciones.
- Diseño de la ruta de datos monociclo
  - Componentes de la ruta de datos. Ensamblaje de la ruta de datos. Ruta de datos monociclo: puntos de control.
- Diseño del controlador monociclo
  - Determinación de los valores de los puntos de control. Control global vs. Control local. Ruta datos monociclo + controlador. Temporización monociclo.
- Diseño de la ruta de datos (multiciclo)
  - Ruta de datos multiciclo: con y sin buses.
- Diseño del controlador (multiciclo)
  - Diagrama de estados del controlador. El controlador como una FSM. Alternativas de implementación del controlador

Temporización (multiciclo)



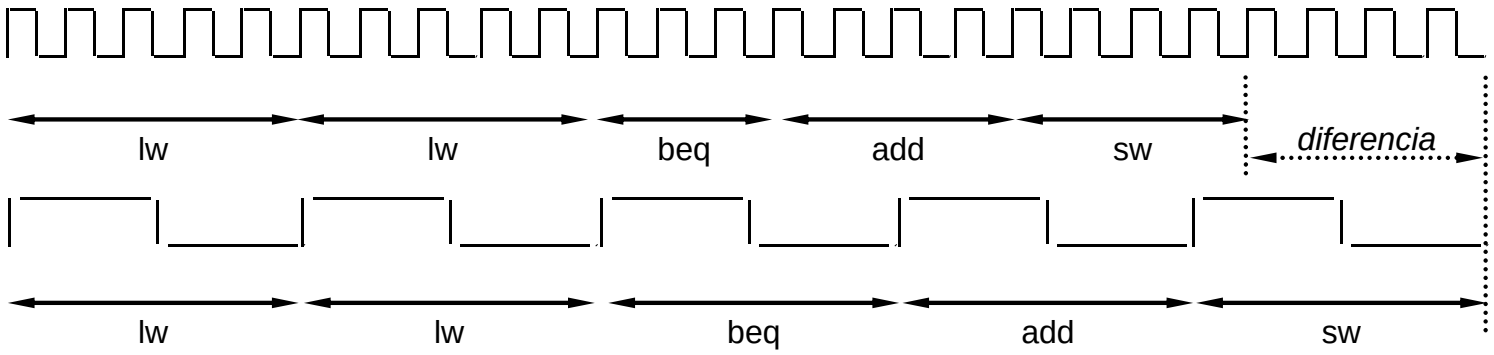
Comparación monociclo vs multiciclo

Asumiendo un comportamiento **ideal**:

- El procesador monociclo tarda 5ns en ejecutar una instrucción
- El procesador multiciclo divide la ejecución en 5 operaciones de 1 ns:
  - IF: Se lee la instrucción de la memoria de instrucciones
  - ID: se decodifica la instrucción y se leen los operandos del banco de registros. Se extiende el signo del operando inmediato
  - EX: Se utiliza la ALU para realizar los cálculos
  - MEM: Se lee o escribe en la memoria de datos
  - WB: Se escribe en banco de registros
- El procesador multiciclo necesita:
  - __ ciclos para instruccions Lw
  - __ ciclos para aritméticas
  - __ ciclos para Sw
  - __ para saltos tomados
  - __ para saltos no tomados

Comparación monociclo vs multiciclo

```
lw $t2, 0($t3)
lw $t3, 4($t3)
beq $t2, $t3, Label #asumir que no se salta
add $t5, $t2, $t3
sw $t5, 8($t3)
```



Comparación: monociclo vs. multiciclo

Operación	*Frecuencia*	*Ciclos*	*CPI*
tipo-R	50%	4	___
lw	20%	5	___
st	10%	4	___
beq (salta)	2.5%	4	___
beq (no salta)	17.5%	3	___
			=

10^6 instrucciones tardan en ejecutarse:

✓ $t_{multi} = 10^6 \cdot CPI_{multi} \cdot t_{multi} = 10^6 \cdot \_\_ \cdot 1ns$

✓ $t_{mono} = 10^6 \cdot CPI_{mono} \cdot t_{mono} = 10^6 \cdot \_\_ \cdot 5ns$

$t_{multi} / t_{mono} = \_\_ / \_\_ = \_\_$

Idealmente los programas tardan un

__ % menos

en ejecutarse en el computador multiciclo

Pero en la práctica:

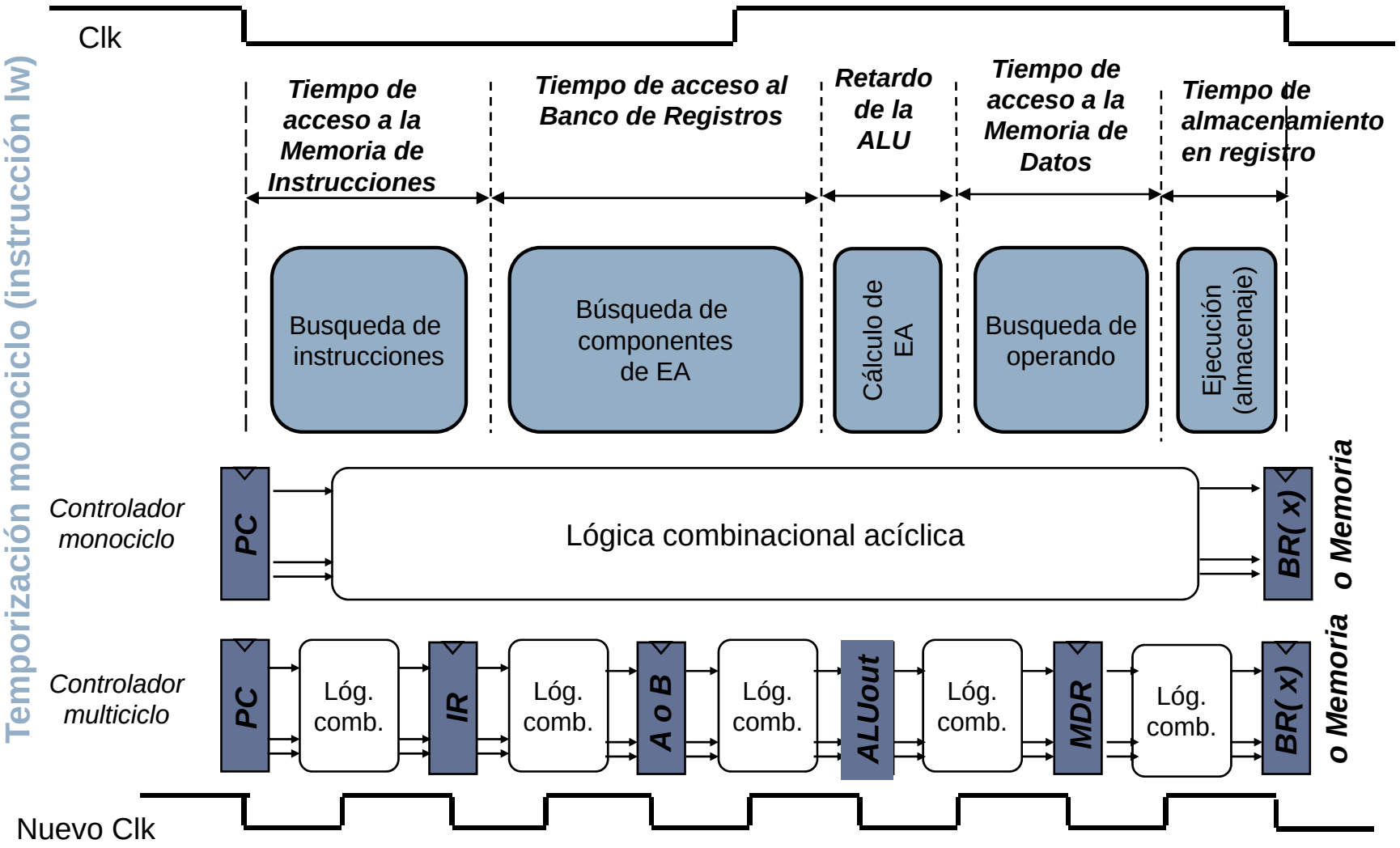
- No es posible dividir la ejecución en 5 etapas iguales
- Algunas etapas tendrán más retardo que otras
  - Reloj final marcado por la etapa más lenta
- Registros intermedios procesador multiciclo: retardos adicionales

Comparación: monociclo vs. multiciclo

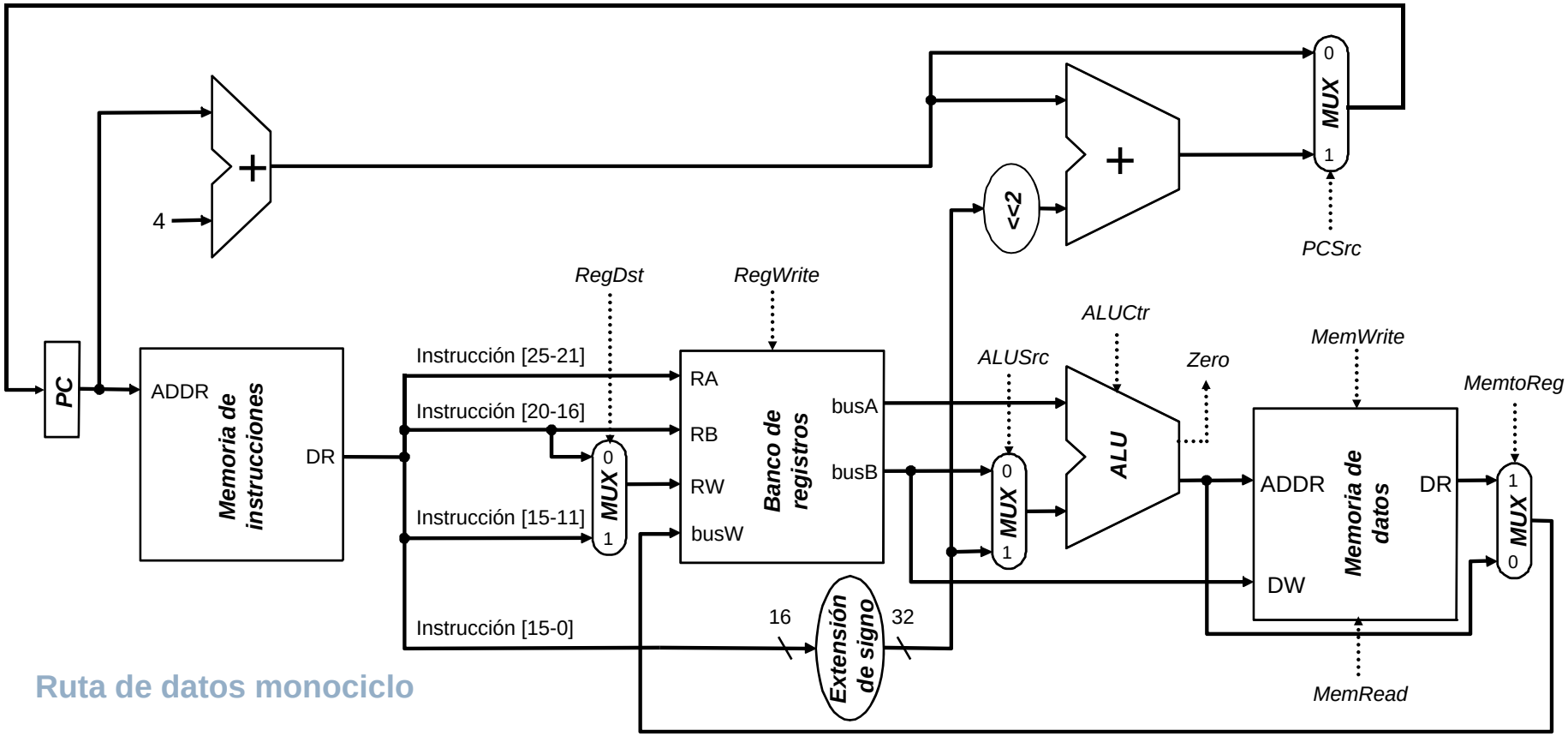
- **Ventajas monociclo:**
  - control más sencillo
  - No se necesitan almacenamientos intermedios
- **Desventajas monociclo:**
  - El reloj debe tener igual periodo que la instrucción más lenta
  - No se puede reutilizar hardware

- **Ventajas multiciclo:**
  - cada instrucción tarda el número de ciclos que sea necesario
  - podemos reutilizar un recurso HW en dos ciclos distintos
- **Desventajas multiciclo:**
  - El reloj debe tener igual periodo que la etapa más lenta
  - Se necesitan registros intermedios para que los datos generados en una etapa puedan usarse en las siguientes, esto conlleva un incremento del coste y del retardo.
  - Aumenta la complejidad del controlador

Temporización (multiciclo)



Diseño de la ruta de datos multiciclo

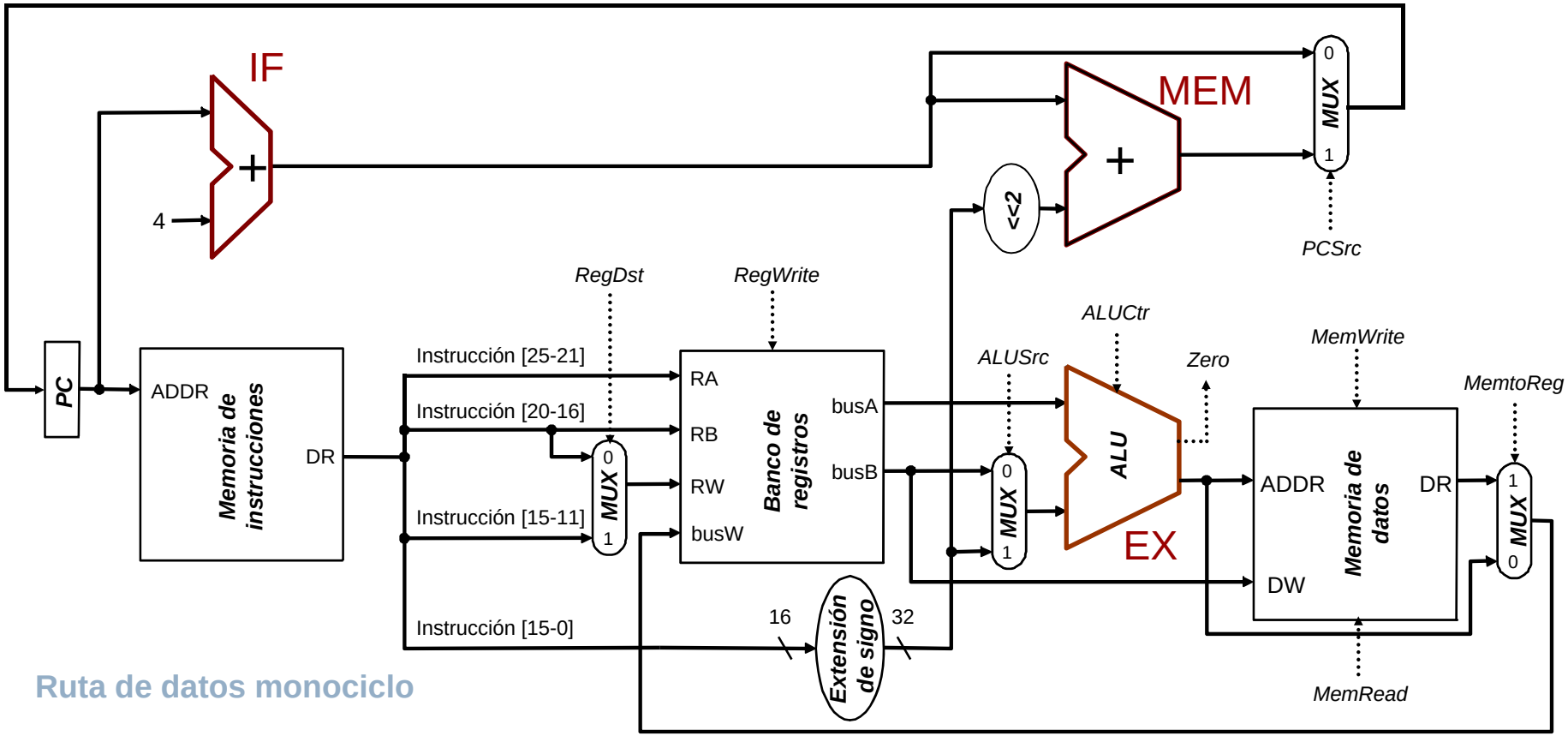


Ruta de datos monociclo

Podemos reutilizar hardware

Hay que guardar todo aquello que se genere en un ciclo y se utilice más adelante

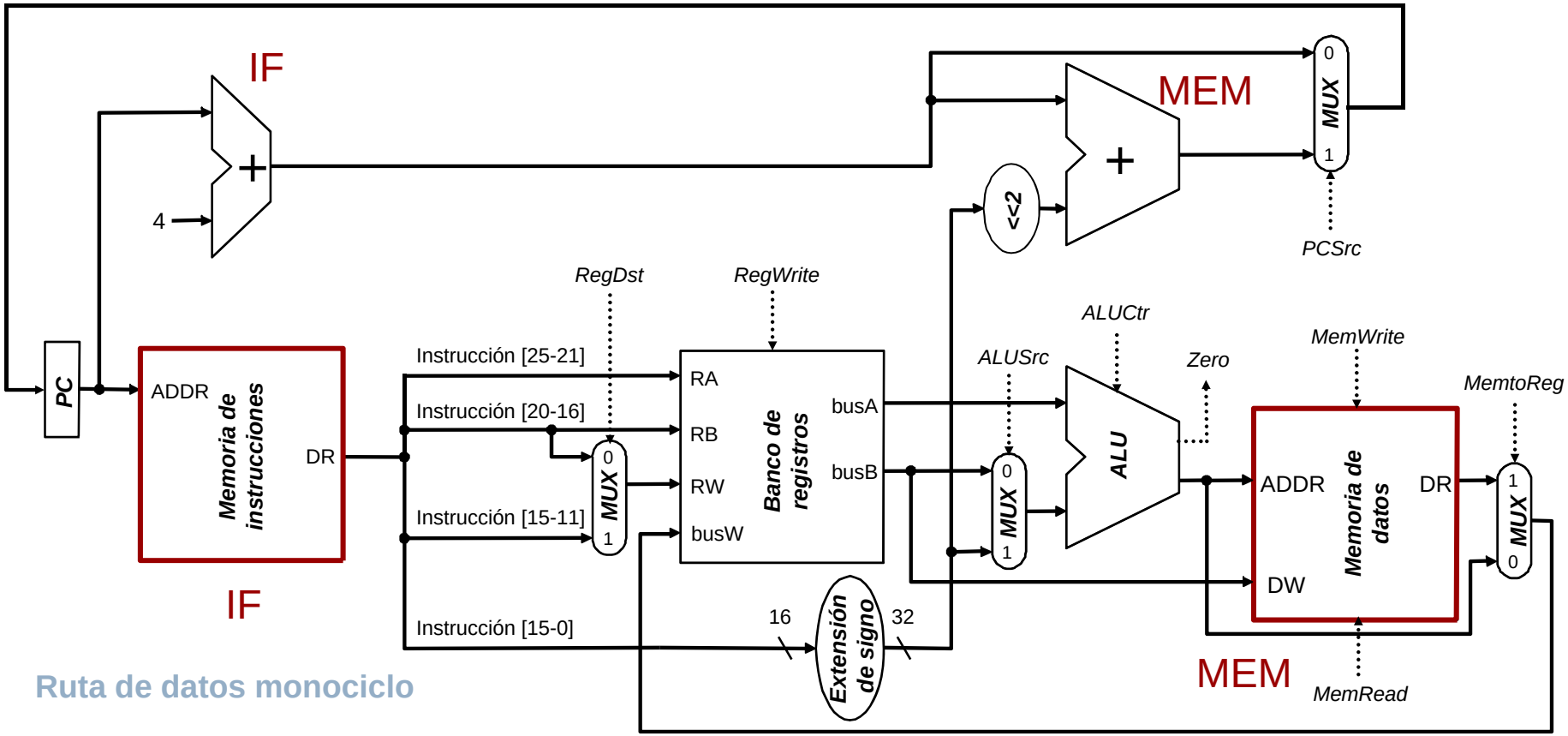
Diseño de la ruta de datos multiciclo: reutilización



Ruta de datos monociclo

Las dos sumas las podría hacer la ALU siempre que se realicen en etapas distintas

Diseño de la ruta de datos multiciclo: reutilización

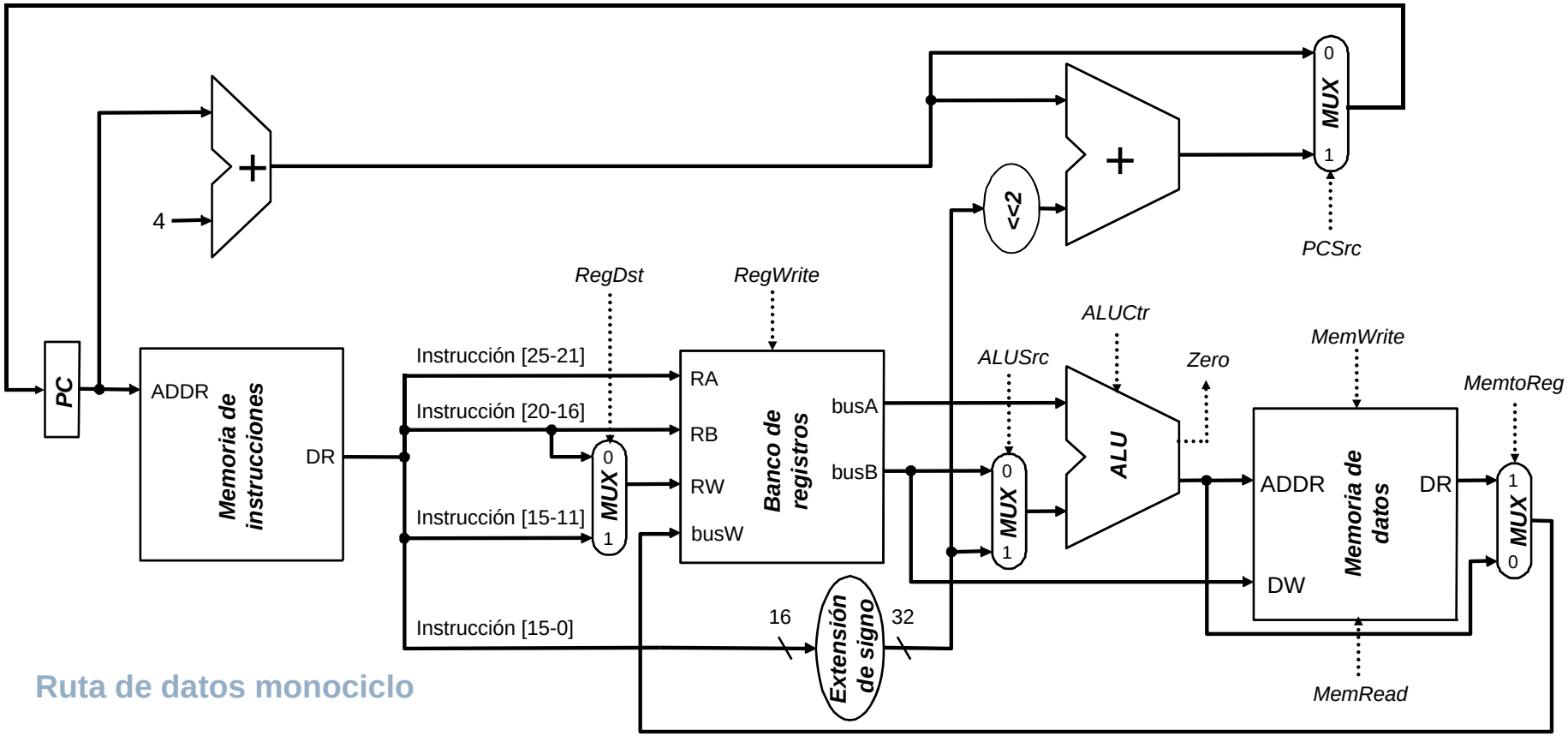


Ruta de datos monociclo

Ya no se accede en el mismo ciclo a la memoria de datos y a la de instrucciones:

Se puede utilizar una única memoria

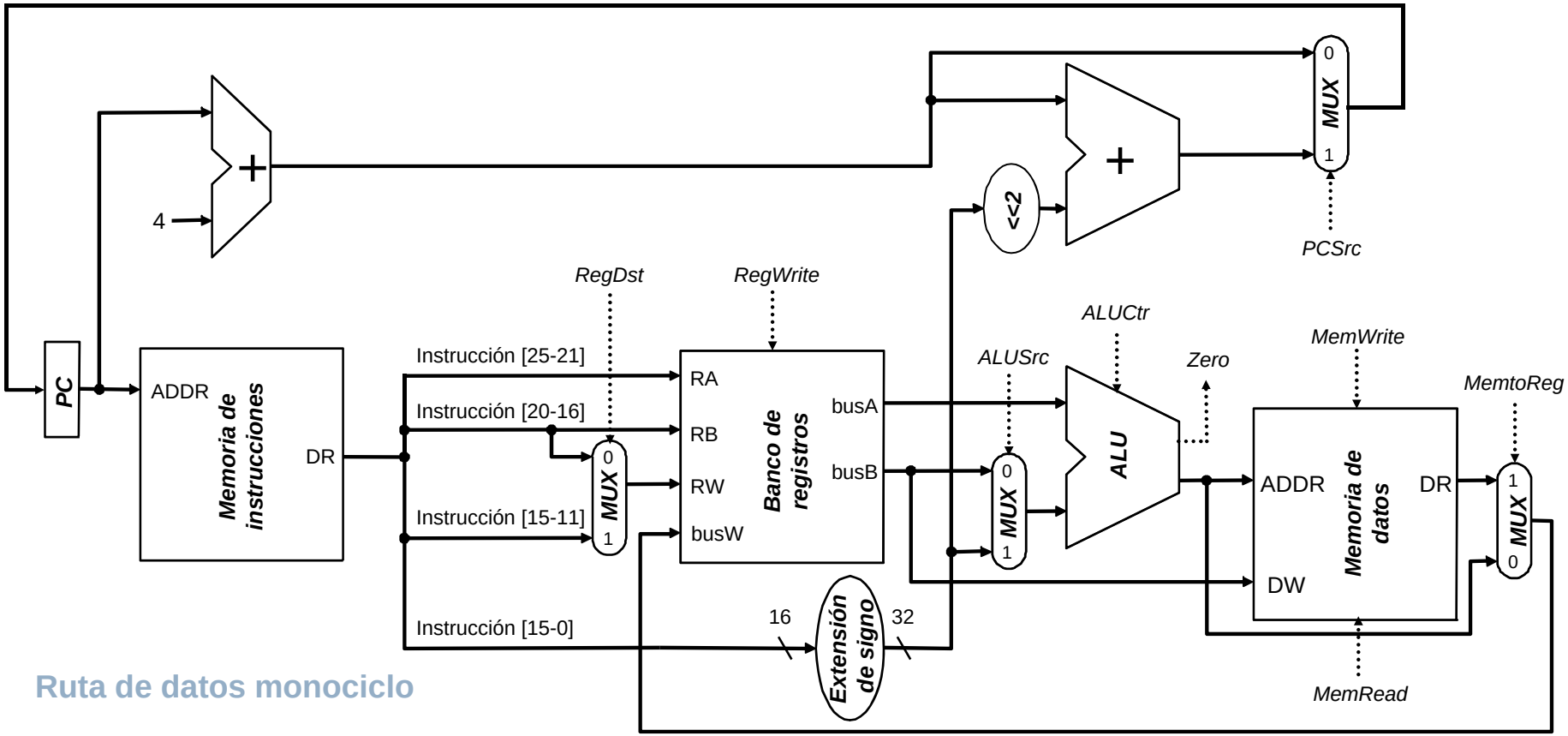
Diseño de la ruta de datos multiciclo: registros intermedios



Ruta de datos monociclo

IF: leemos la instrucción de la memoria y hay que guardarla para para utilizarla en las etapas posteriores: registro IR (instruction register)

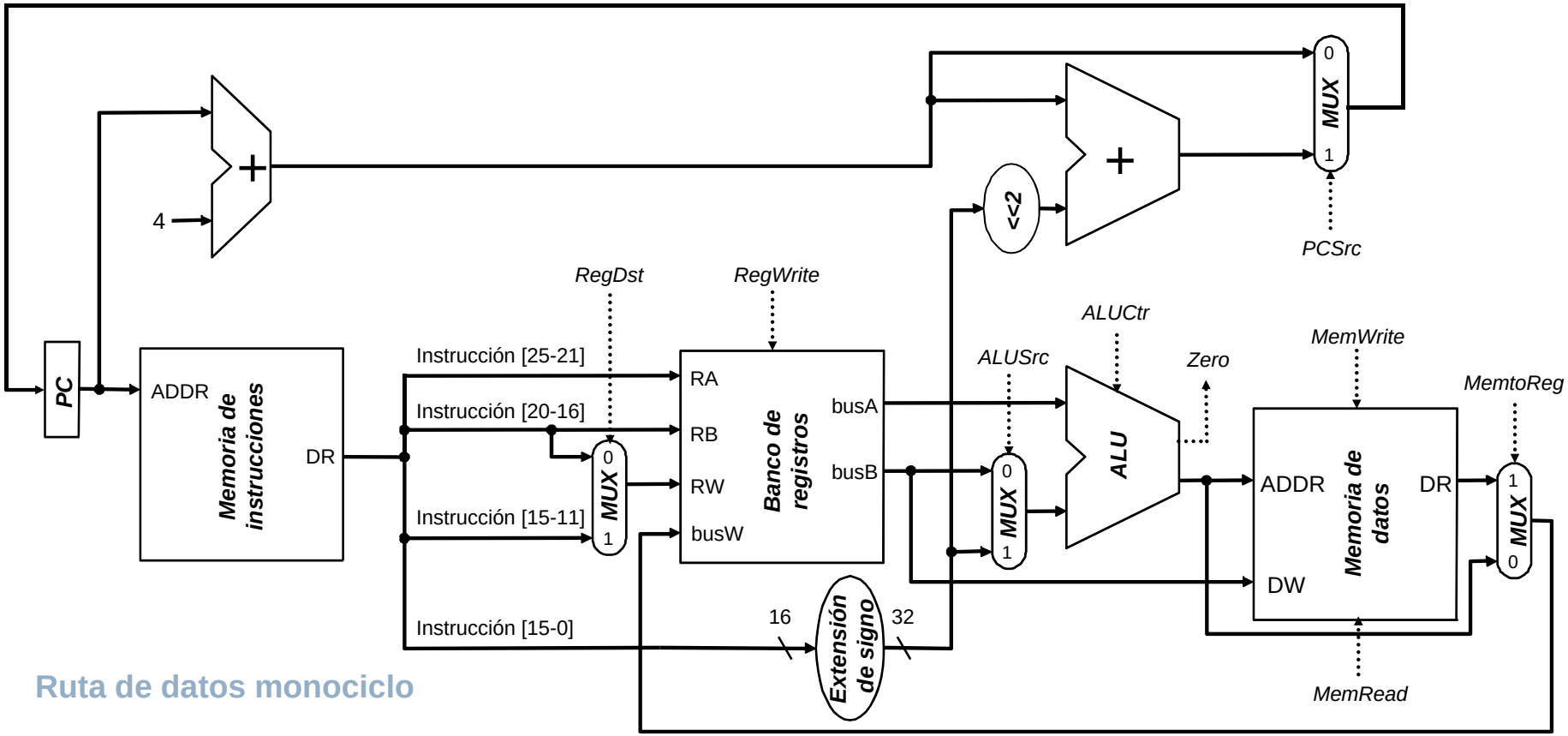
Diseño de la ruta de datos multiciclo: registros intermedios



Ruta de datos monociclo

ID: leemos los operandos del banco de registros que se utilizarán en la etapa EX y MEM: registros A y B

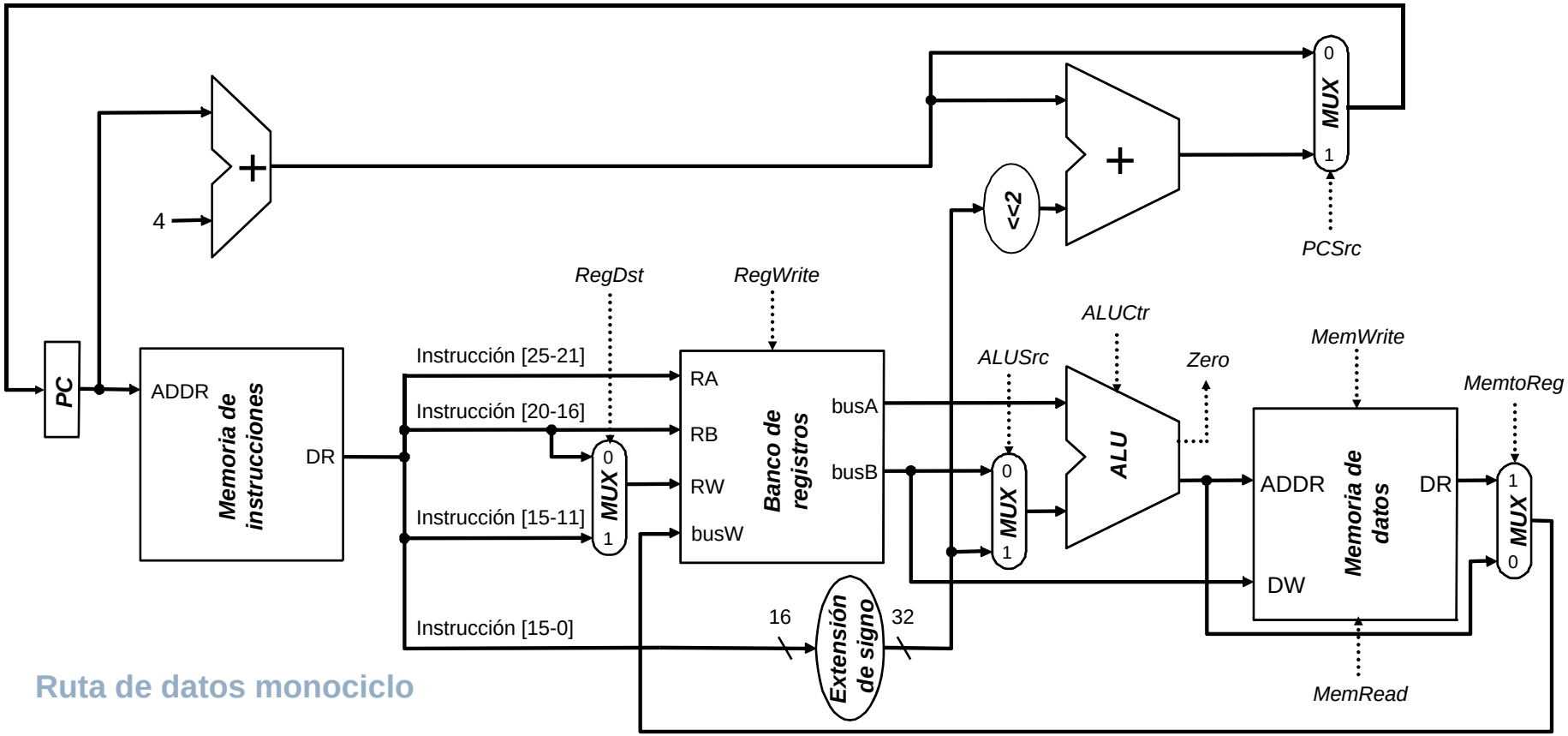
Diseño de la ruta de datos multiciclo: registros intermedios



Ruta de datos monociclo

EX: Se realizan las operaciones con la ALU, el resultado se usa en los siguientes ciclos: registro ALUout

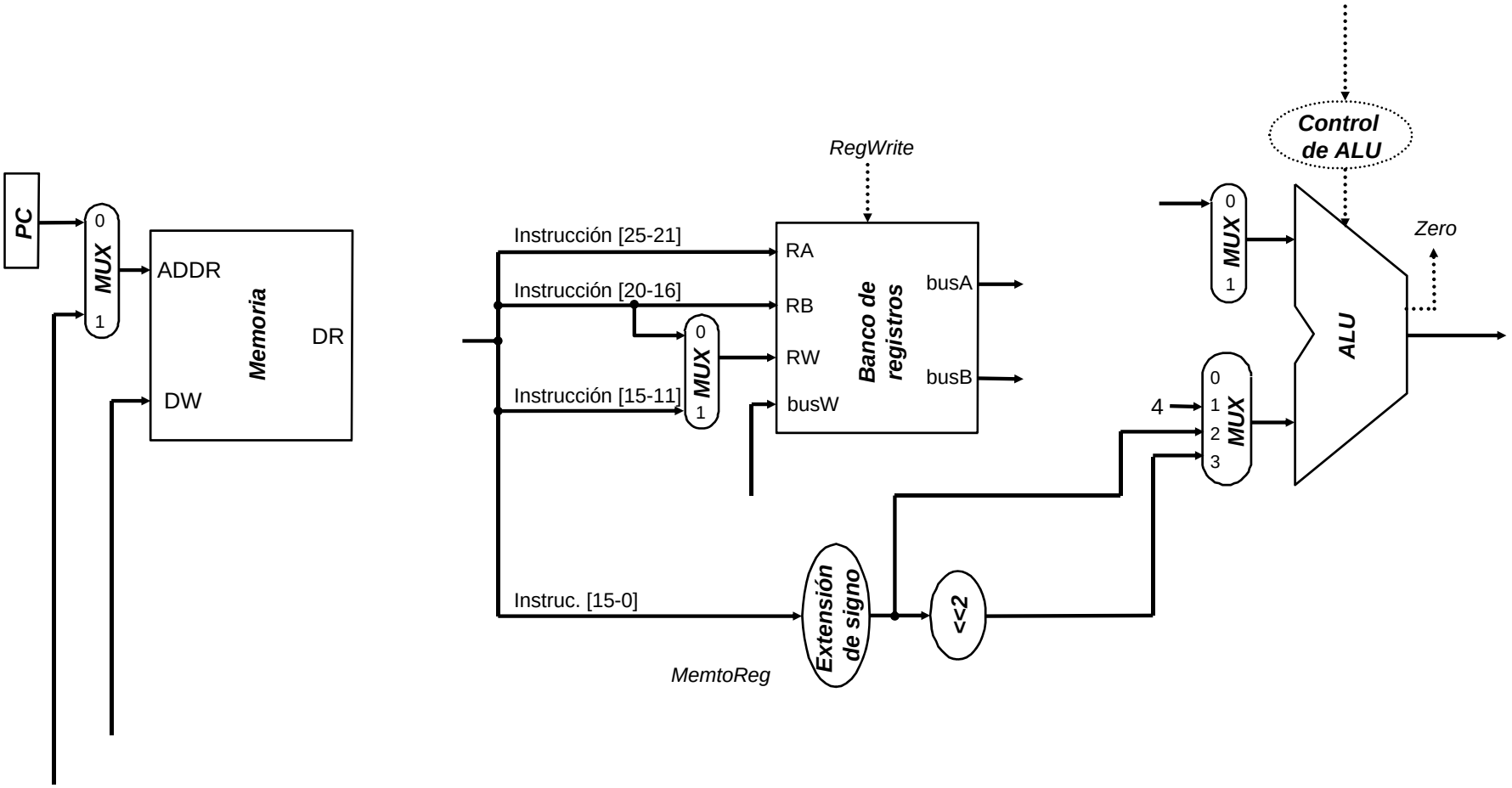
Diseño de la ruta de datos multiciclo: registros intermedios



Ruta de datos monociclo

MEM: en las instrucciones lw leemos un dato de memoria que se guardará en el ciclo siguiente en el banco de registros: registro MDR (memory data register)

Diseño de la ruta de datos (multiciclo)



Ruta de datos multiciclo

Diseño del controlador (multiciclo)

Transferencias entre registros “lógicas”

BR(rt) ← Memoria(BR(rs) + SignExt(inmed)), PC ← PC + 4

Transferencias entre registros “físicas”

1. _____ ← Memoria(PC), PC ← PC + 4
2. _____ ← BR(rs)
3. _____ ← A + SignExt(inmed)
4. _____ ← Memoria(ALUout)
5. BR(rt) ← _____

Instrucción de carga (lw)

Transferencias entre registros “lógicas”

Memoria(BR(rs) + SignExt(inmed)) ← BR(rt), PC ← PC + 4

Transferencias entre registros “físicas”

1. _____ ← Memoria(PC), PC ← PC + 4
2. _____ ← BR(rs), _____ ← BR(rt)
3. _____ ← A + SignExt(inmed)
4. Memoria(ALUout) ← B

Instrucción de almacenaje (sw)

Diseño del controlador (multiciclo)

Transferencias entre registros “lógicas”

BR(rd) ← BR(rs) funct BR(rt), PC ← PC + 4

Transferencias entre registros “físicas”

1. _____ ← Memoria(PC), PC ← PC + 4
2. _____ ← BR(rs), _____ ← BR(rt)
3. _____ ← _____ funct _____
4. BR(rd) ← _____

Instrucción aritmética (tipo-R)

Transferencias entre registros “lógicas”

si (BR(rs) = BR(rt)) entonces PC ← PC + 4 + 4*SignExt(inmed)

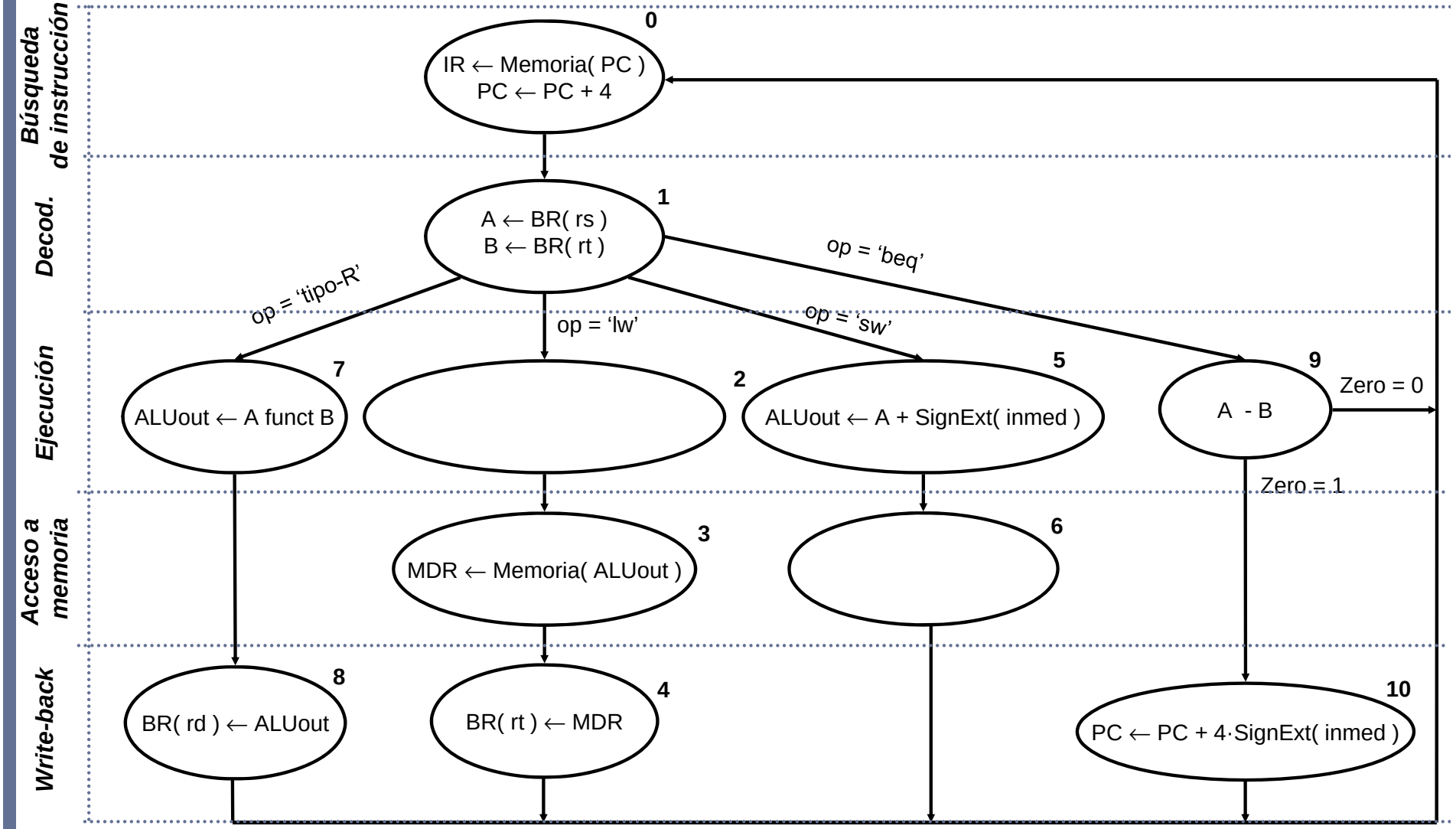
sino PC ← PC + 4

Transferencias entre registros “físicas”

1. _____ ← Memoria(PC), PC ← PC + 4
2. _____ ← BR(rs), _____ ← BR(rt),
3. _____ - _____
4. si Zero entonces PC ← PC + 4·SignExt(inmed)

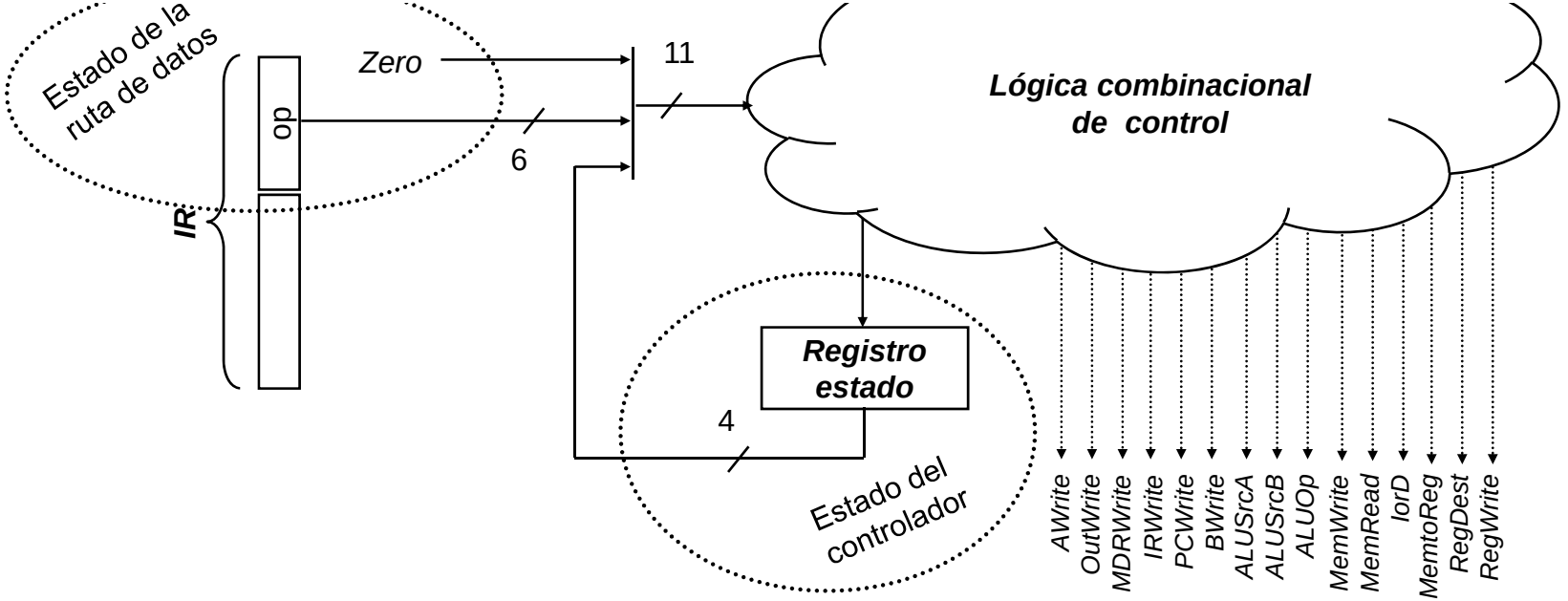
Instrucción de salto condicional (beq)

Diseño del controlador (multiciclo)



Diseño del controlador (multiciclo)

1. Se **traducen** las transferencias entre registros como **conjuntos de activaciones** de los puntos de control de la ruta de datos
2. Se **codifican** los estados
3. Mediante **tablas de verdad** se describen:
   - ✓ las **transiciones de estado** en función del código de operación y del estado de la ruta de datos
   - ✓ el **valor de las señales** de control en función del estado (controlador tipo Moore) y adicionalmente en función del estado de la ruta de datos (controlador tipo Mealy).
4. La **estructura del controlador** estará formada por:
   - ✓ **Registro de estado**
   - ✓ Conjunto de **lógica combinacional de control** que implementa las anteriores tablas



Conclusiones

- El procesador monociclo no es adecuado para repertorios de instrucciones muy heterogéneos
- El procesador multiciclo puede mejorar sensiblemente el rendimiento con respecto al monociclo porque:
  - permite utilizar un reloj más rápido
  - y adaptar el tiempo que de ejecución de cada instrucción a sus necesidades:
    - Instrucciones sencillas se ejecutan en pocos ciclos
    - Instrucciones complejas requieren un número de ciclos elevado
- Pero las ganancias obtenidas suelen estar lejos de las máximas teóricas porqué:
  - El reloj debe tener igual periodo que la etapa más lenta
  - Se necesitan registros intermedios que incrementan el retardo.
- En cuanto al área:
  - Permite reutilizar elementos hardware (como la ALU)
  - Para ello normalmente hay que añadir algunos multiplexores
  - Por otro lado es necesario añadir almacenamiento intermedio para todos los datos que se obtienen en un ciclo y se usan más adelante
  - La unidad de control es más compleja