Arquitectura de Computadoras para Ingeniería

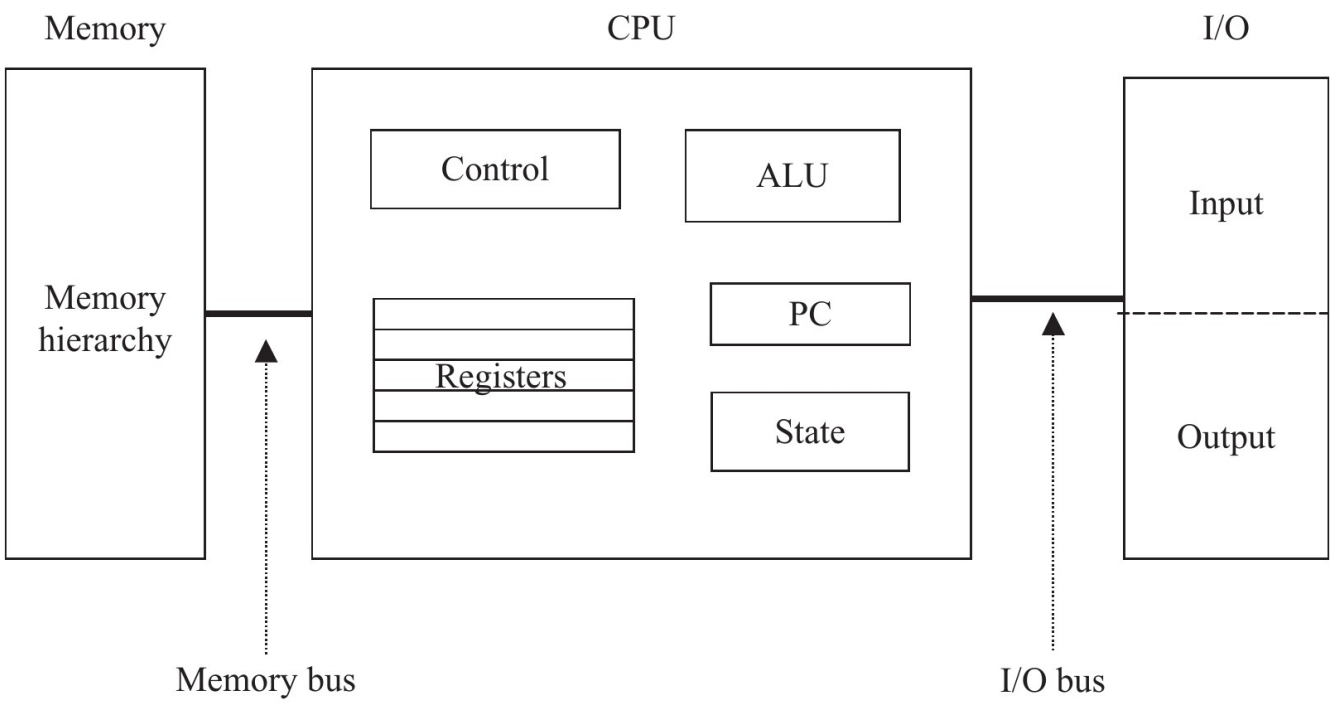
(Cód. 7526)
1° Cuatrimestre 2016

Dra. Dana K. Urribarri
DCIC - UNS

Modelos de arquitecturas

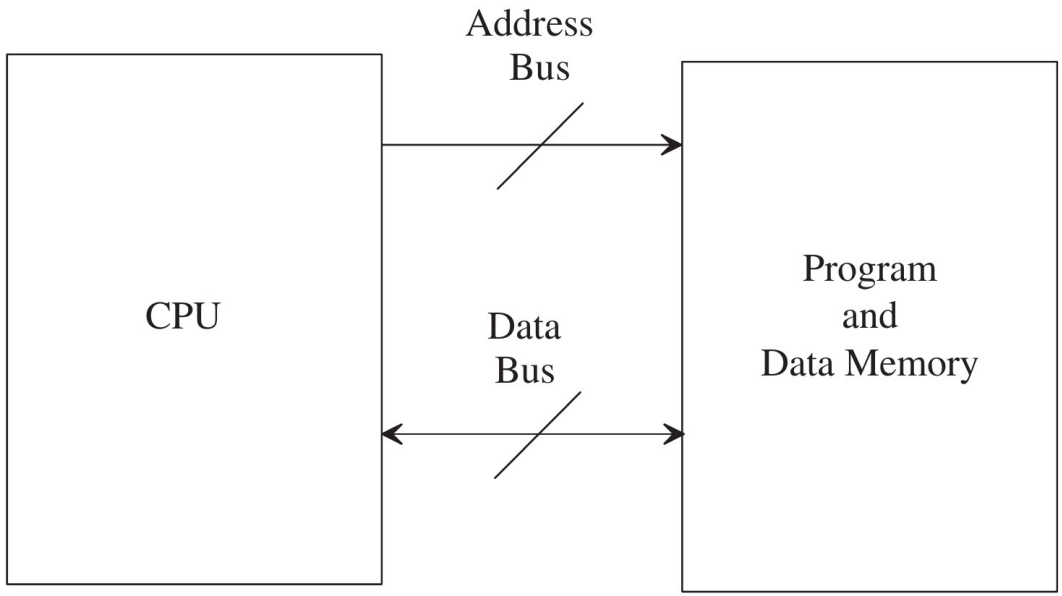
Modelo von Neumann

- El modelo von Neumann tiene 4 partes:
  1) Unidad central de proceso (CPU)
  2) Memoria
  3) Entradas
  4) Salidas



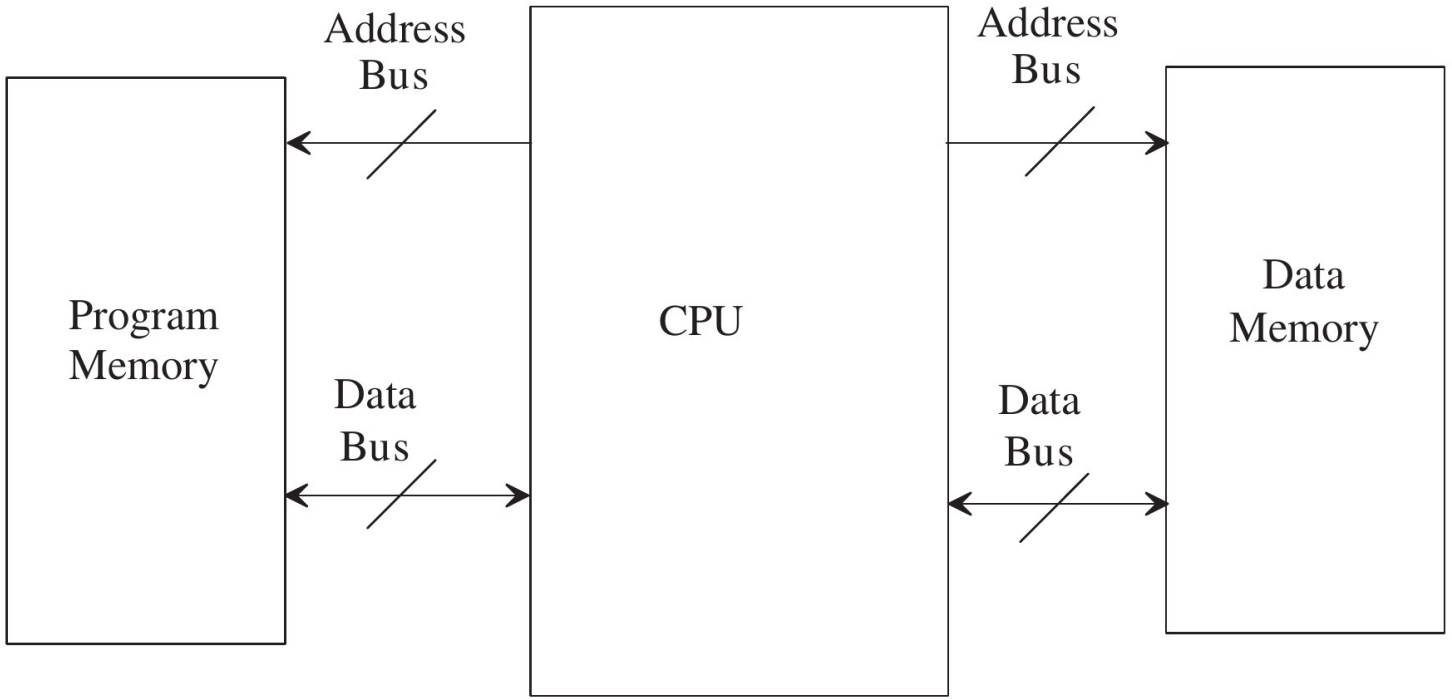
Modelo von Neumann

1) CPU:
   - ALU
   - Registros de alta velocidad para almacenar los operandos
   - Unidad de control: interpreta las instrucciones y hace que se ejecuten
   - Program counter (PC)
2) Memoria
   - Almacena instrucciones, datos y resultados intermedios y finales. Se implementa como una jerarquía.



Modelo Harvard

- Usa memoria y buses separados para instrucciones y datos.
- Pueden ejecutar instrucciones y acceder a datos simultáneamente.
- Requiere 4 buses, dos de direcciones y dos de contenido.



Microprocesador vs Microcontrolador

- Microprocesador:
  - Chip que contiene la CPU.
  - Modelo von Neumann. La memoria es externa al microprocesador.

- Microcontrolador:
  - Un único chip que contiene CPU, memoria, entrada/salida y periféricos.
  - Modelo Harvard.

Medidas de desempeño

Medidas de desempeño

- Para comparar arquitecturas se necesita mediciones que reflejen:
  - la arquitectura del procesador
  - la jerarquía de memoria

  independientemente:
  - de los programas
  - del subsistema de I/O.

Medidas de desempeño

- Ciclos por instrucción (CPI)
- Instrucciones por ciclo (IPC)
- Rendimiento (*Throughput*)
- Latencia
- *Speedup*
- Eficiencia

Tiempo de ejecución

- El tiempo de ejecución de un programa (EX_{CPU}) depende de la cantidad de instrucciones a ejecutar y del tiempo de ejecución de cada instrucción.
- El tiempo de ejecución de una instrucción puede medirse en función de la cantidad de ciclos que demora en ejecutarse y del período de cada pulso.

$$EX_{CPU} = \underbrace{\#\text{Instrucciones} \times \text{CPI}}_{\#\text{Total de ciclos}} \times \underbrace{\text{período del reloj}}_{1/\text{frecuencia}}$$

Instrucciones por ciclo

- La recíproca del CPI

$$\text{CPI} = EX_{\text{CPU}} \,/\, (\#\text{Instrucciones x período del reloj})$$

es la cantidad de instrucciones que se ejecutan en un ciclo

$$\text{IPC} = 1 \,/\, \text{CPI}$$

$$\text{IPC} = \#\text{Instrucciones} \times \text{período del reloj} \,/\, EX_{\text{CPU}}$$

$$\text{IPC} = \#\text{Instrucciones} \,/\, \#\text{Total de ciclos}$$

Medidas de desempeño

- Throughput: IPC
  - Cantidad de trabajo (instrucciones) por unidad de tiempo
- Latencia: EX_{CPU}
  - Las unidades de tiempo que demanda realizar el trabajo.

Medidas de desempeño

- El rendimiento (*performance*) se define como la recíproca del tiempo de ejecución.
- Los tres factores que afecta el rendimiento son:
  - La cantidad de instrucciones ejecutadas
    ➞ Responsabilidad del compilador
  - CPI (o IPC)
    ➞ Responsabilidad del diseño e implementación de la arquitectura
  - Frecuencia del reloj
    ➞ Dependiente de la tecnología

Comparación de desempeño

- Comparación en términos relativos (no absolutos)
- Un sistema A tiene mejor rendimiento que un sistema B si el sistema A tiene menor tiempo de ejecución (para un conjunto de programas) que el sistema B.

$$\frac{Rendimiento_A}{Rendimiento_B} = \frac{1/EX_{CPU\,A}}{1/EX_{CPU\,B}} = \frac{EX_{CPU\,B}}{EX_{CPU\,A}}$$

Comparación de desempeño

- El *speedup* es la razón entre el rendimiento de un sistema mejorado y el rendimiento de su implementación original

$$Speedup = \frac{Rendimiento\ Mejorado}{Rendimiento\ Original} = \frac{EX_{CPU\ Original}}{EX_{CPU\ Mejorado}}$$

Eficiencia

- Mientras el speedup es la ganancia por mejorar un sistema.
- La eficiencia mide la utilización de un recurso.

- Si $Speedup_n$ es la ganancia por mejorar el sistema con n recursos, la eficiencia mide la utilización de esos recursos.

$$Eficiencia = \frac{Speedup_n}{n}$$

Ley de Amdahl

- El *speedup* se definió inicialmente para procesadores paralelos.

- Si T_1 es el tiempo de ejecución en un procesador y T_n es el tiempo de ejecución en *n* procesadores, lo esperable sería que:

$$T_n = T_1 / n$$

y por lo tanto

$$Speedup = \frac{T_1}{T_n} = \frac{T_1}{T_1/n} = n$$

Ley de Amdahl

- La ejecución de un programa consiste de una parte *secuencial* seguido de una parte *paralelizable*.
- Si
  - *s* es el tiempo de ejecución de la parte secuencial
  - *p* es el tiempo de ejecución secuencial de la parte paralelizable
- Entonces

$$T_1 = s + p$$

$$T_n = s + p / n$$

$$Speedup = \frac{T_1}{T_n} = \frac{s+p}{s+p/n}$$

Ley de Amdahl

- Si

$$Speedup = \frac{T_1}{T_n} = \frac{s+p}{s+p/n}$$

- En el límite ($n \rightarrow \infty$)

$$\lim_{n\rightarrow\infty} Speedup = \frac{s+p}{s}$$

- Luego, si asumimos $T_1 = 1$ (o 100%) el máximo *speedup* 1/s está limitado por la fracción de tiempo que se consume en la ejecución de la parte secuencial.

Microarquitectura de microprocesador

(Modelo von Neumann)

Arquitectura y Microarquitectura

- La *arquitectura* de una computadora la define:
  - El set de instrucciones
  - La ubicación de los operandos (registros y memoria)
  - x86, MIPS, SPARC, PowerPC.

- La arquitectura no define la implementación del hardware.
  - Intel y AMD implementan la arquitectura x86.
  - Diferencias en rendimiento, precio y consumo.

- La *microarquitectura* es la combinación específica de registros, memoria, ALUs y otros bloques del microprocesador.

Arquitectura

- Vamos a considerar DLX, un subconjunto de MIPS.
- MIPS:
  - RISC
  - 32 y 64 bits
- Implementaciones de MIPS se usan en Sistemas Embebidos:
  - Dispositivos con Windows CE
  - Routers
  - Consolas como Nintendo 64, Sony PlayStation, PlayStation 2 y PlayStation Portable.

Arquitectura DLX

DLX incluye:

- **Instrucciones R-type**

  Instrucciones aritmético-lógico

- **Instrucciones de escritura y lectura en memoria**

  Load y store

- **Instrucciones de bifurcación**

  Saltos condicionles (branch) e incondicionales (jump)

Microarquitectura

- La microarquitectura se divide en dos partes que interactúan entre sí:
  - Datapath
  - Unidad de Control
- Datapath
  - Opera sobre los datos.
  - Involucra memorias, registros, ALUs y multiplexores.
- Unidad de Control
  - A partir de la instrucción actual le indica al datapath cómo ejecutar la instrucción.
  - Produce señales de multiplexado, habilitaciones de registros y señales a la memoria para controlar las operaciones del datapath.

Ciclo de ejecución

Los pasos básicos para ejecutar una instrucción:

1) Se trae de memoria la próxima instrucción a ejecutar (apuntada por el PC)
2) La unidad de control decodifica la instrucción
3) Se ejecuta la instrucción. Puede ser:
   - Operación de la ALU
   - Cargar un registro con un dato de memoria
   - Almacenar un dato de un registro en memoria
   - Testear la condición de un salto
4) Se actualiza el PC
5) Volver al paso 1

Diseño de la microarquitectura

- Tres posibles microarquitecturas para MIPS:
  - **Diseño único ciclo:**

    Ejecuta la instrucción completa en un único ciclo.
  - **Diseño multi-ciclo:**

    Ejecuta las instrucciones en una serie de ciclos cortos.
  - **Diseño en pipeline:**

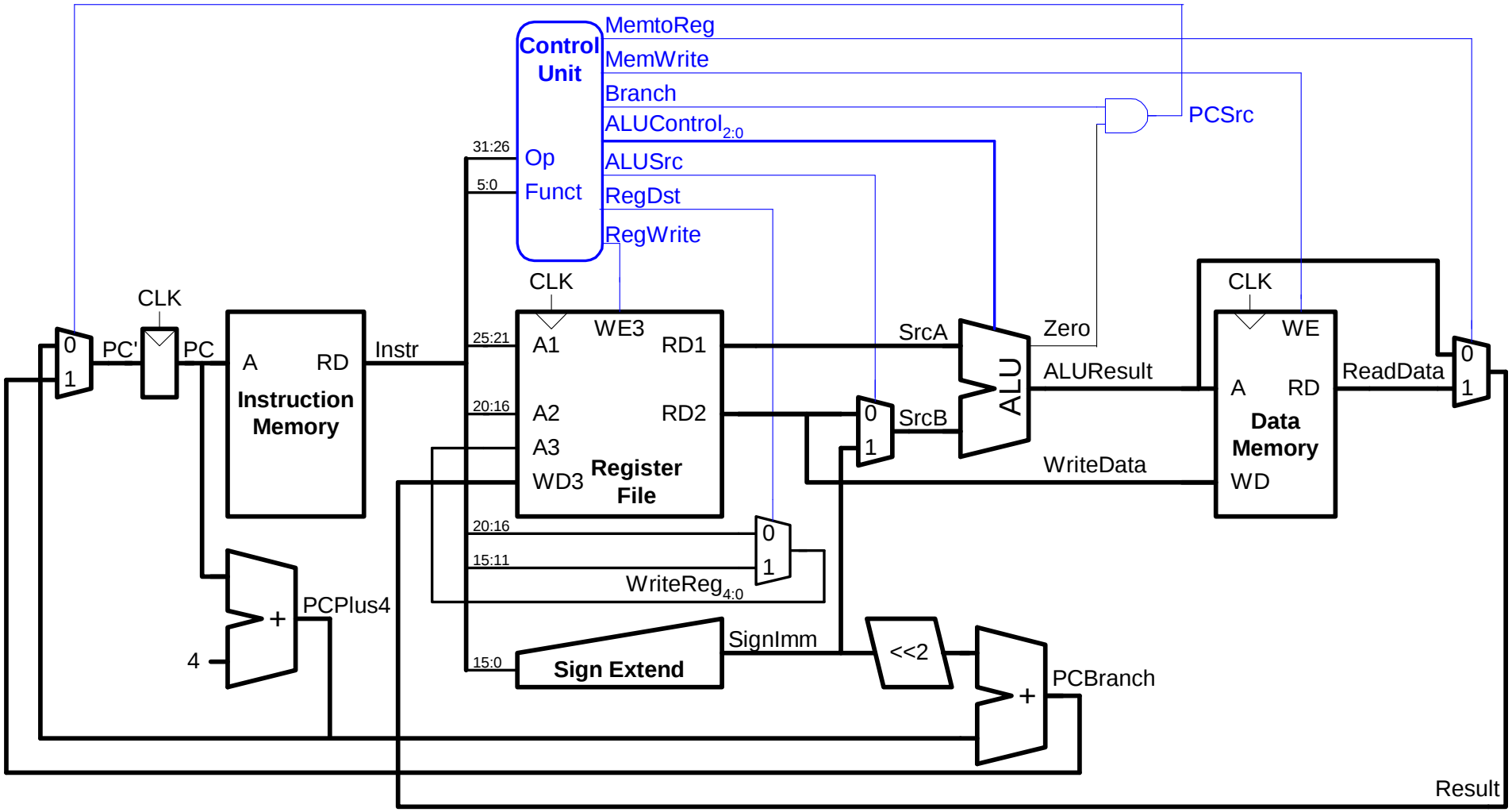
    Aplica pipeline a la microarquitectura de un único ciclo.

Microarquitectura único ciclo

- Cada instrucción se ejecuta en un único ciclo de reloj (CPI = 1).
  - Comienza en un flanco ascendente (o descendente)
  - Termina en el próximo flanco ascendente (o descendente)
- El ciclo tiene que ser lo suficientemente largo para permitir la ejecución de la instrucción más lenta.

- No es práctico pero es simple.

Microarquitectura único ciclo



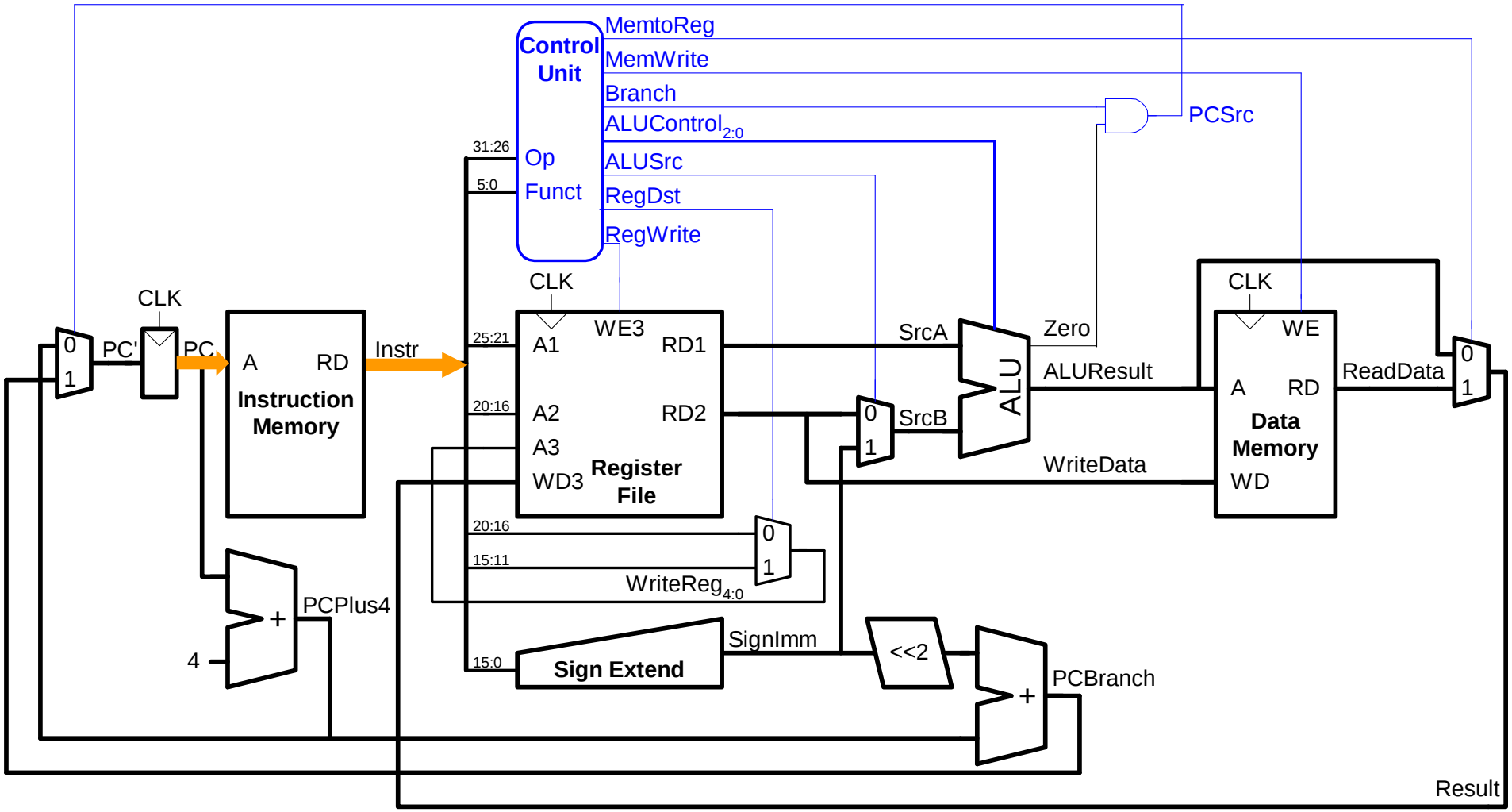
Microarquitectura único ciclo

Load:

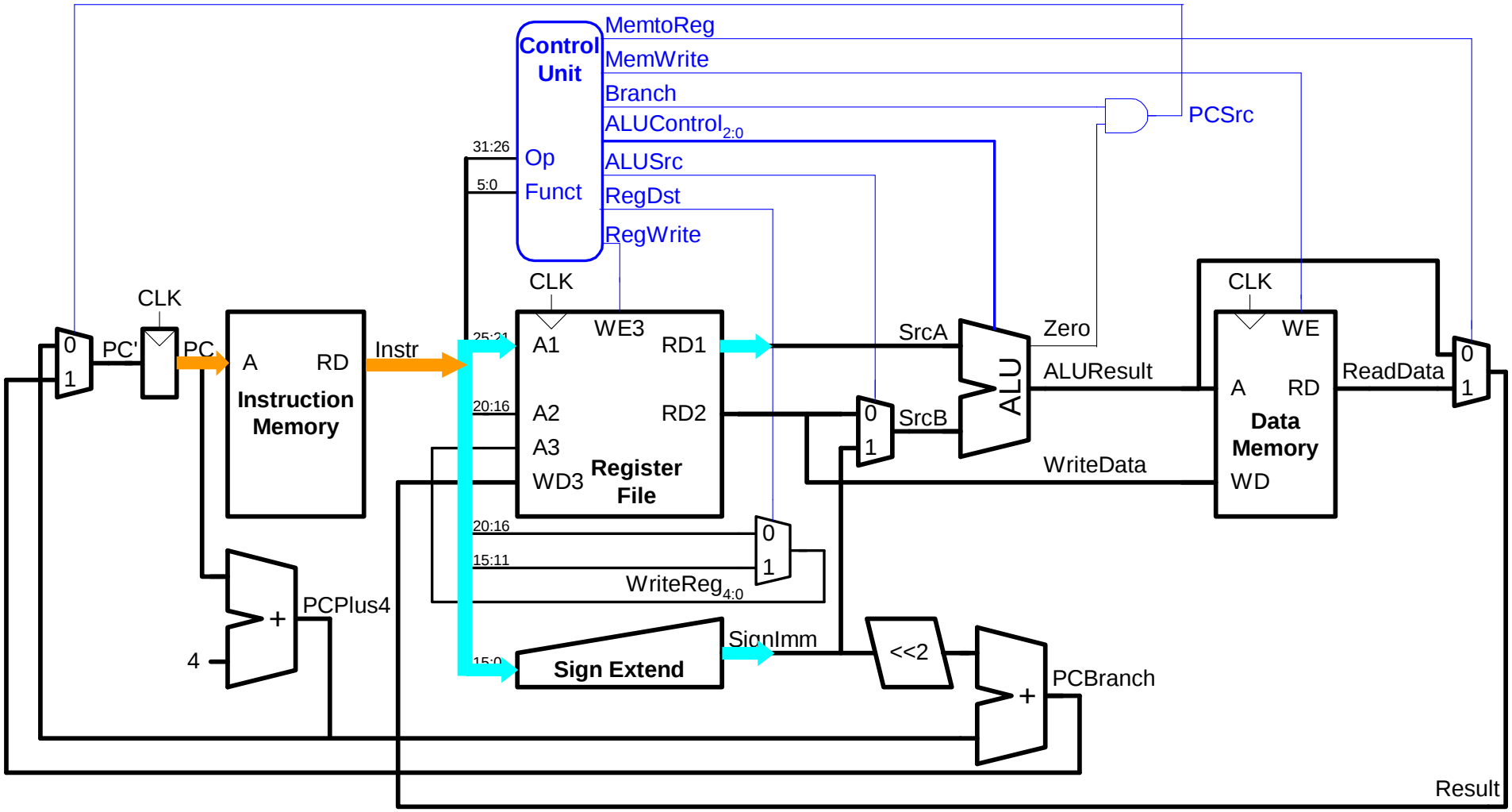
- R[d] ← Mem[R[s] + #valor]
- PC ← PC + 4

1) Traer la instrucción
2) Obtener los operandos: leer el registro del banco de registros y obtener el valor inmediato
3) Calcular la dirección de memoria
4) Leer el dato de memoria y escribirlo en el banco de registros.
5) Actualizar el PC

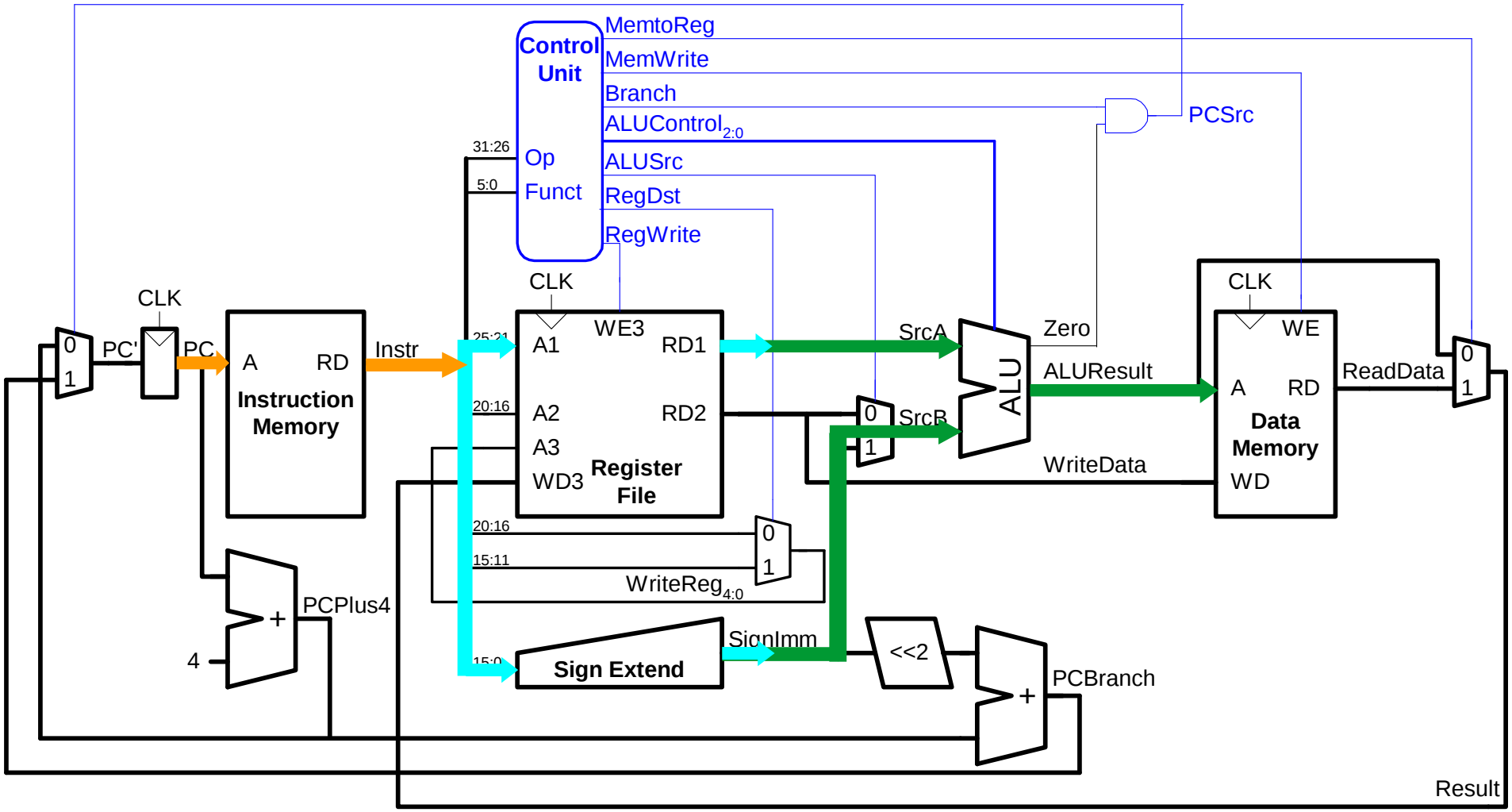
Load: Traer la instrucción



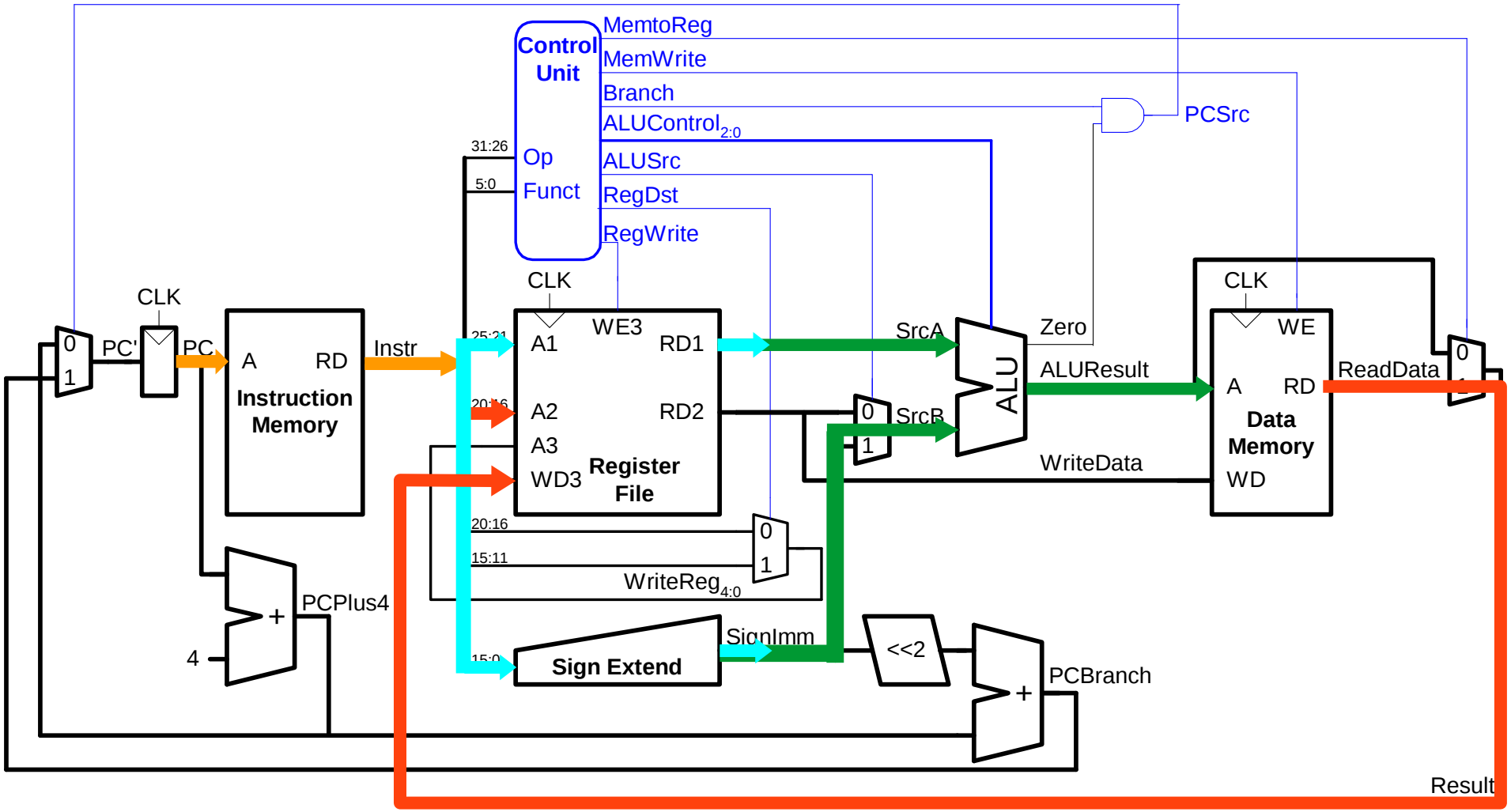
Load: Obtención de operandos



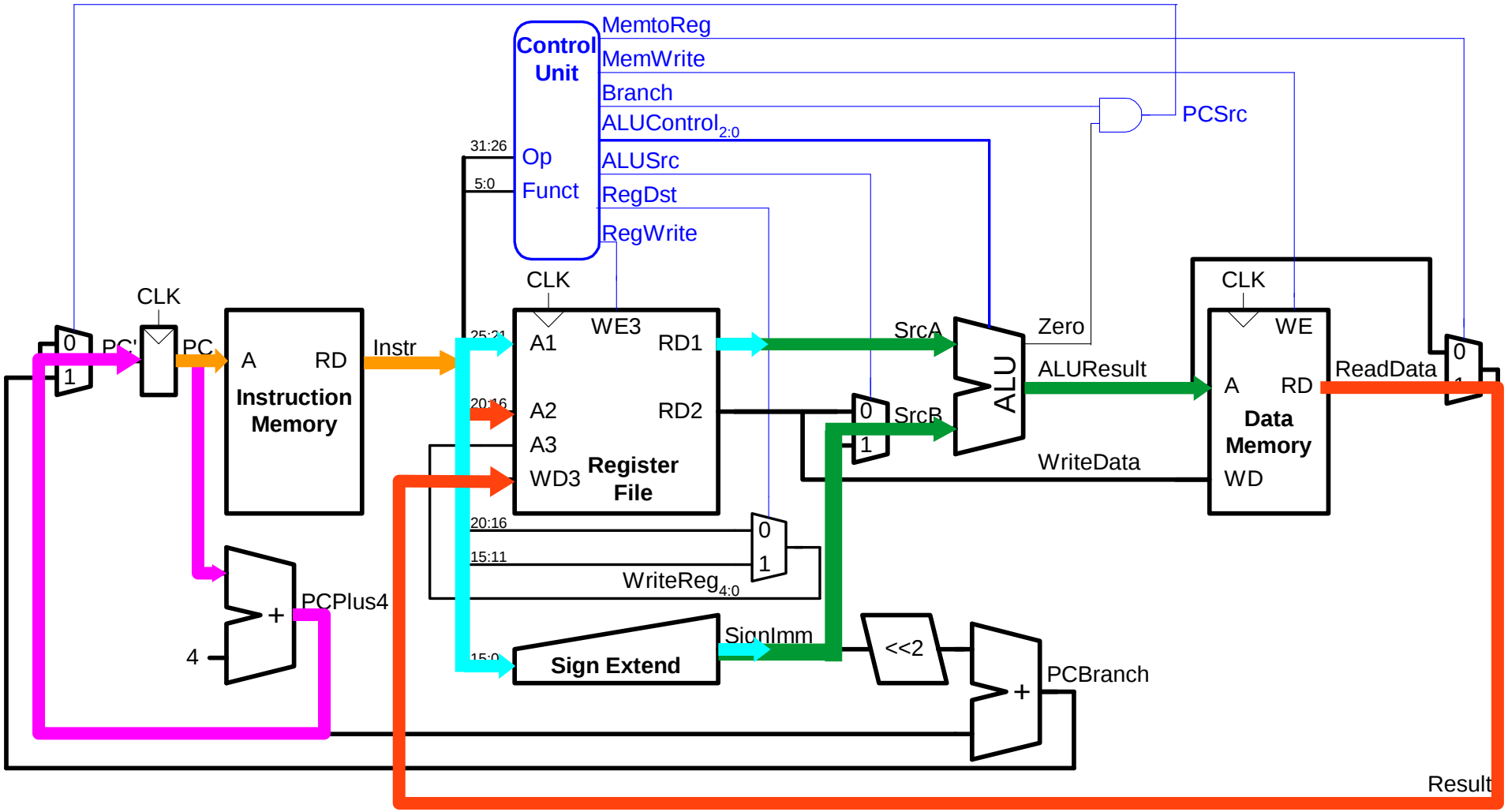
Load: Cálculo de dirección



Load: Escribir el dato obtenido en el destino



Load: Actualizar PC



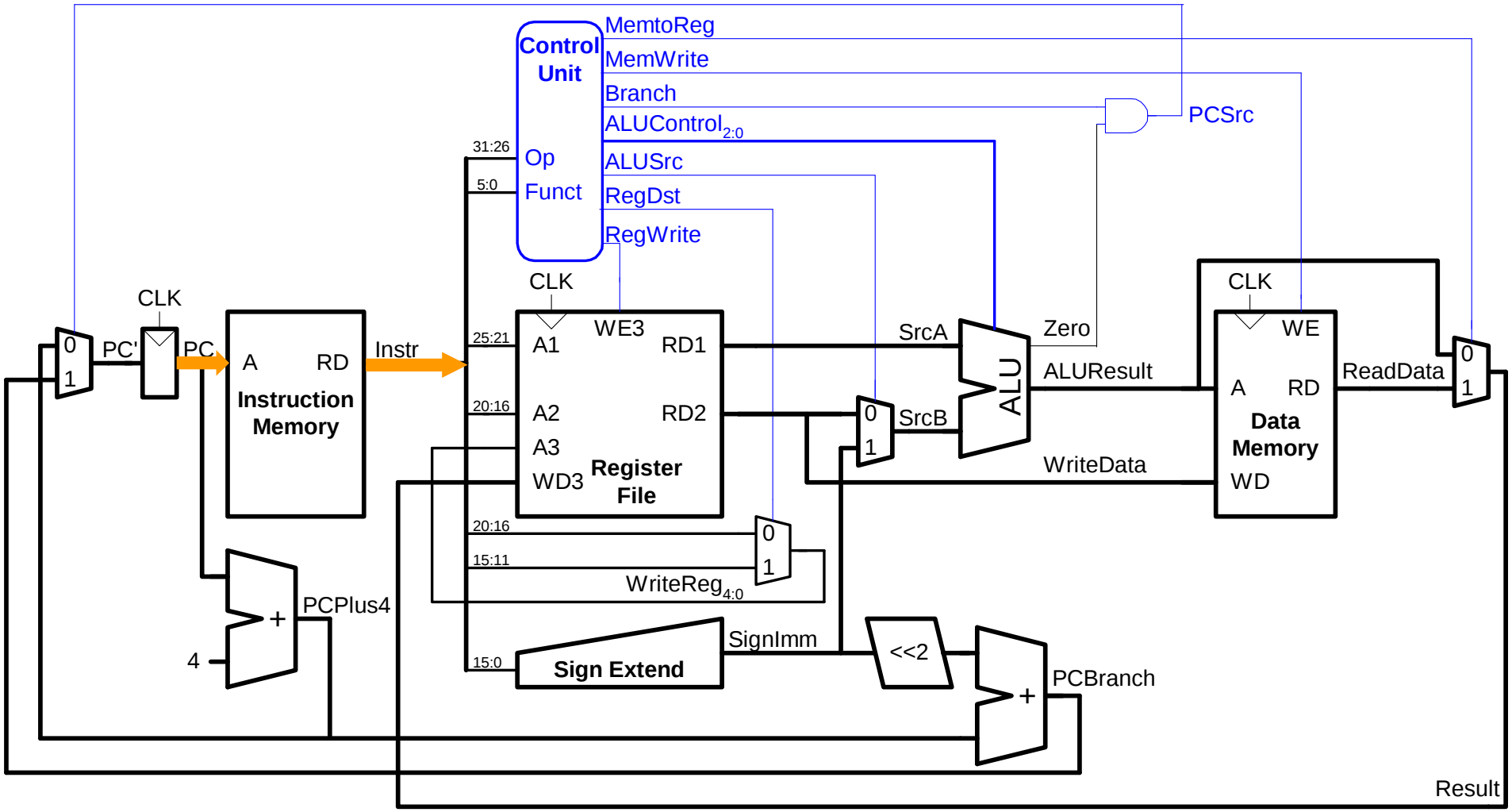
Microarquitectura único ciclo

R-type (add, or, etc):

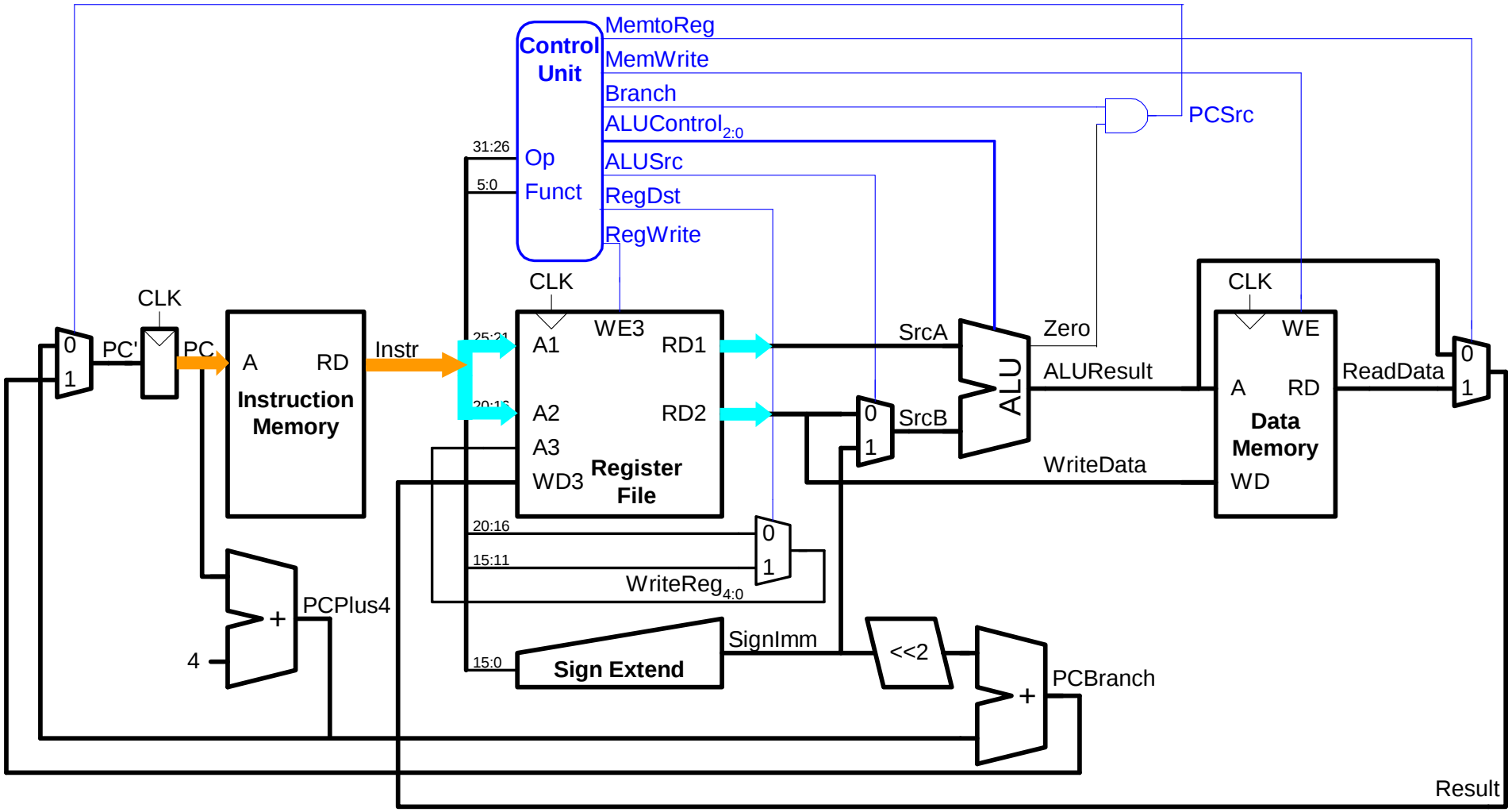
- R[d] ← R[s] + R[t]
- PC ← PC + 4

1) Traer la instrucción
2) Obtener los operandos: leer el registro del banco de registros los dos operandos
3) Realizar la operación de la ALU
4) Escribir el resultado en el banco de registros.
5) Actualizar el PC

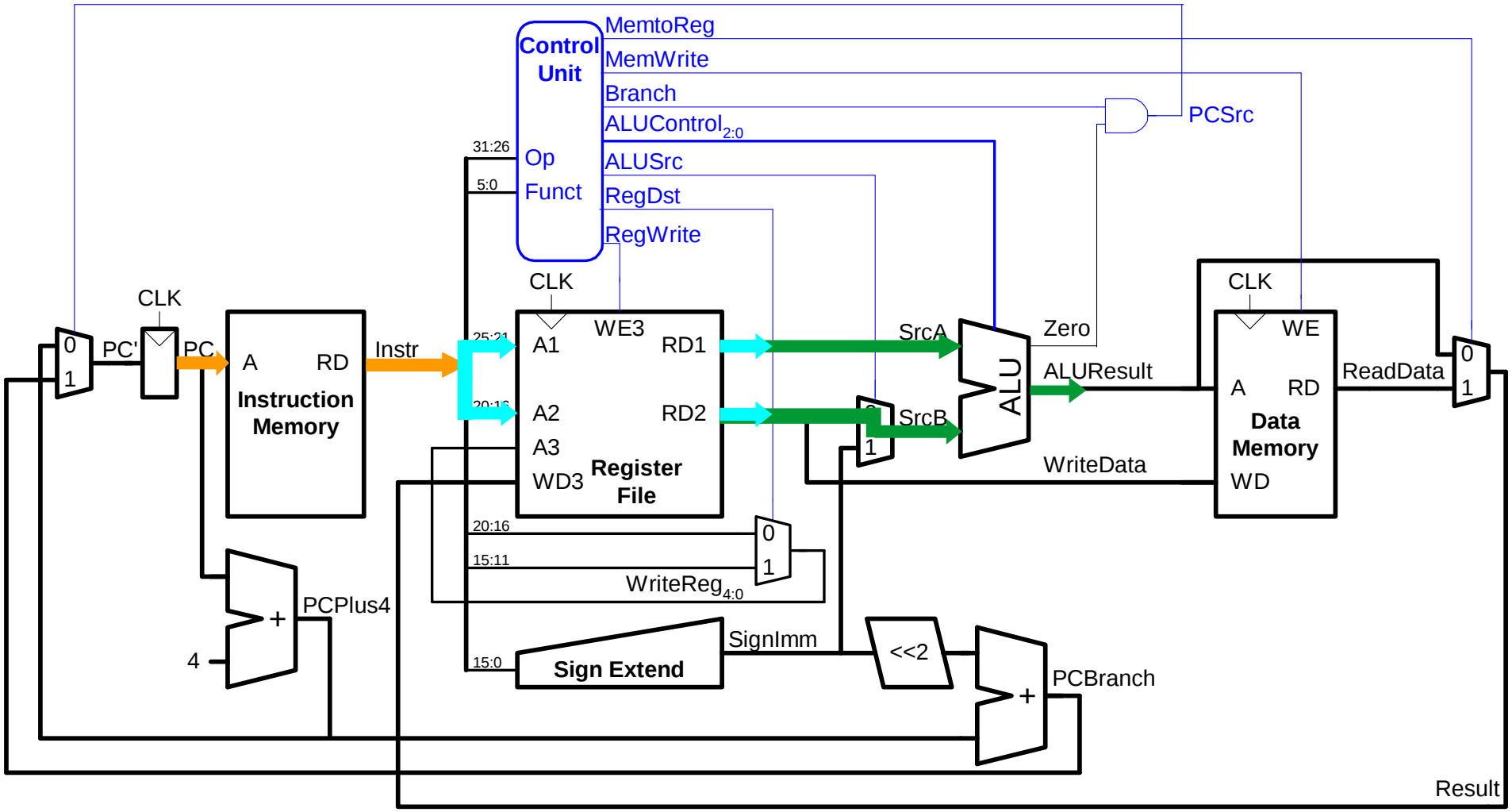
R-type: Traer la instrucción



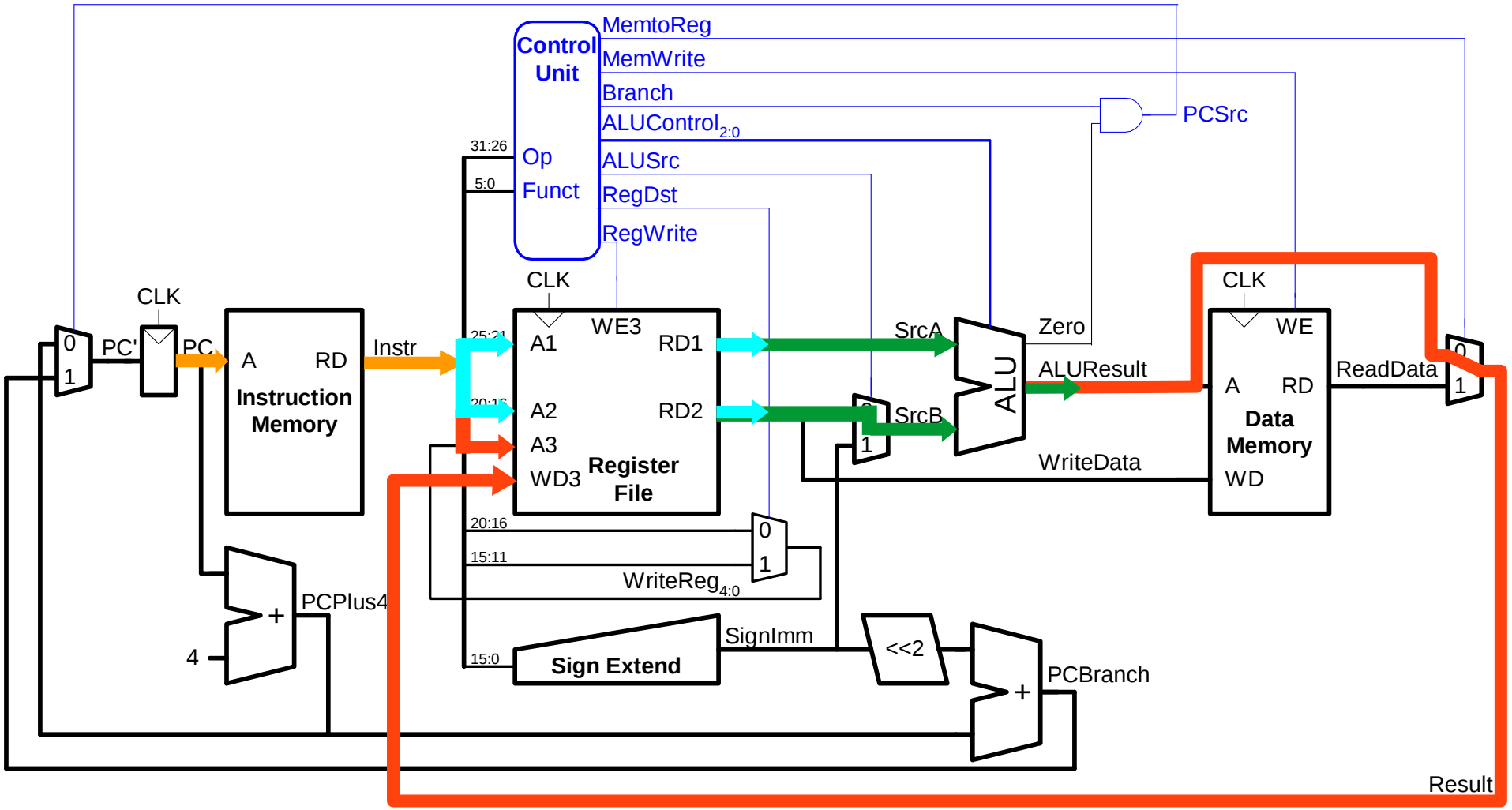
R-type: Obtención de operandos



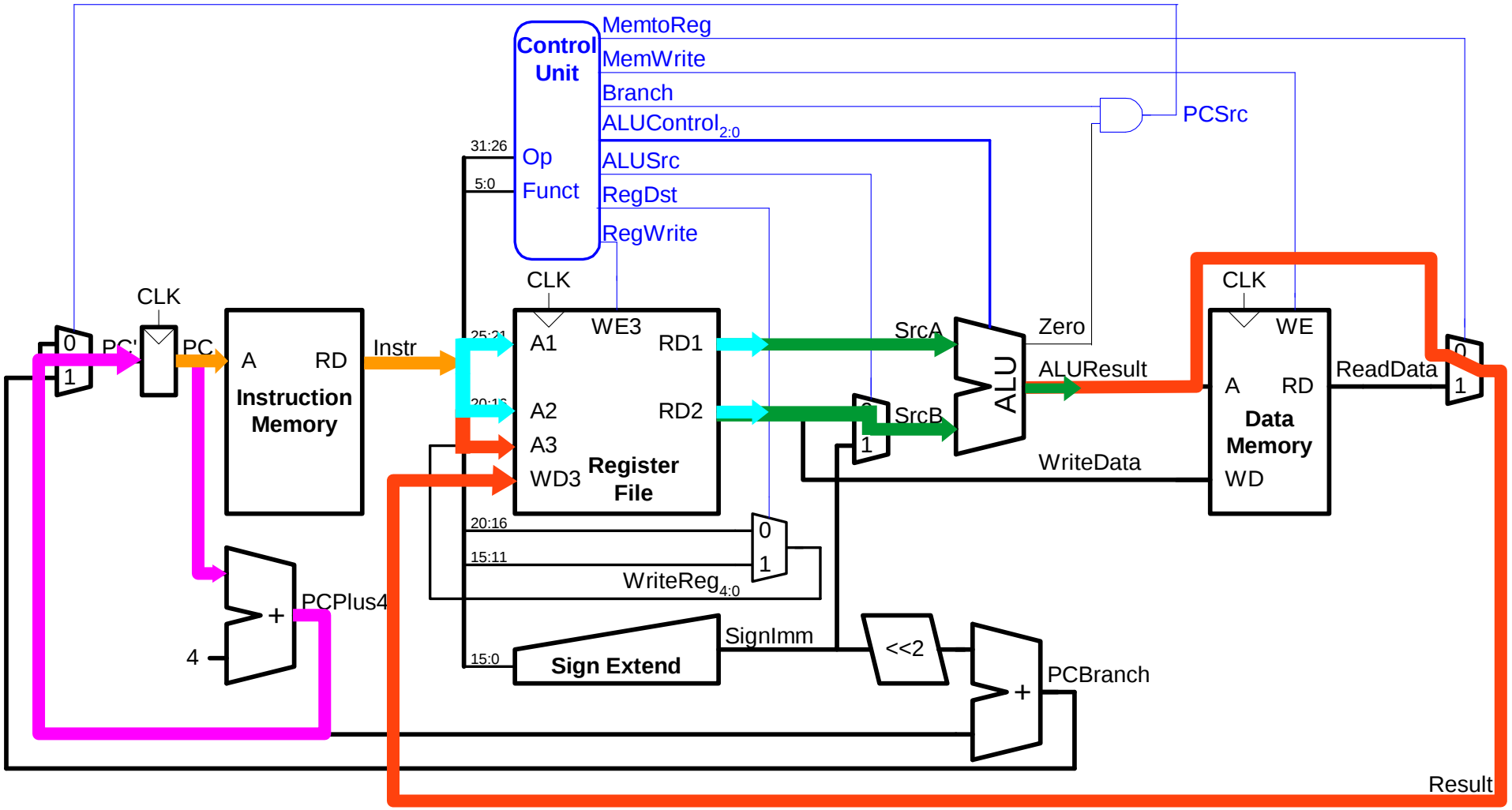
R-type: Realizar la operación de la ALU



R-type: Escribir el resultado



R-type: Actualizar PC



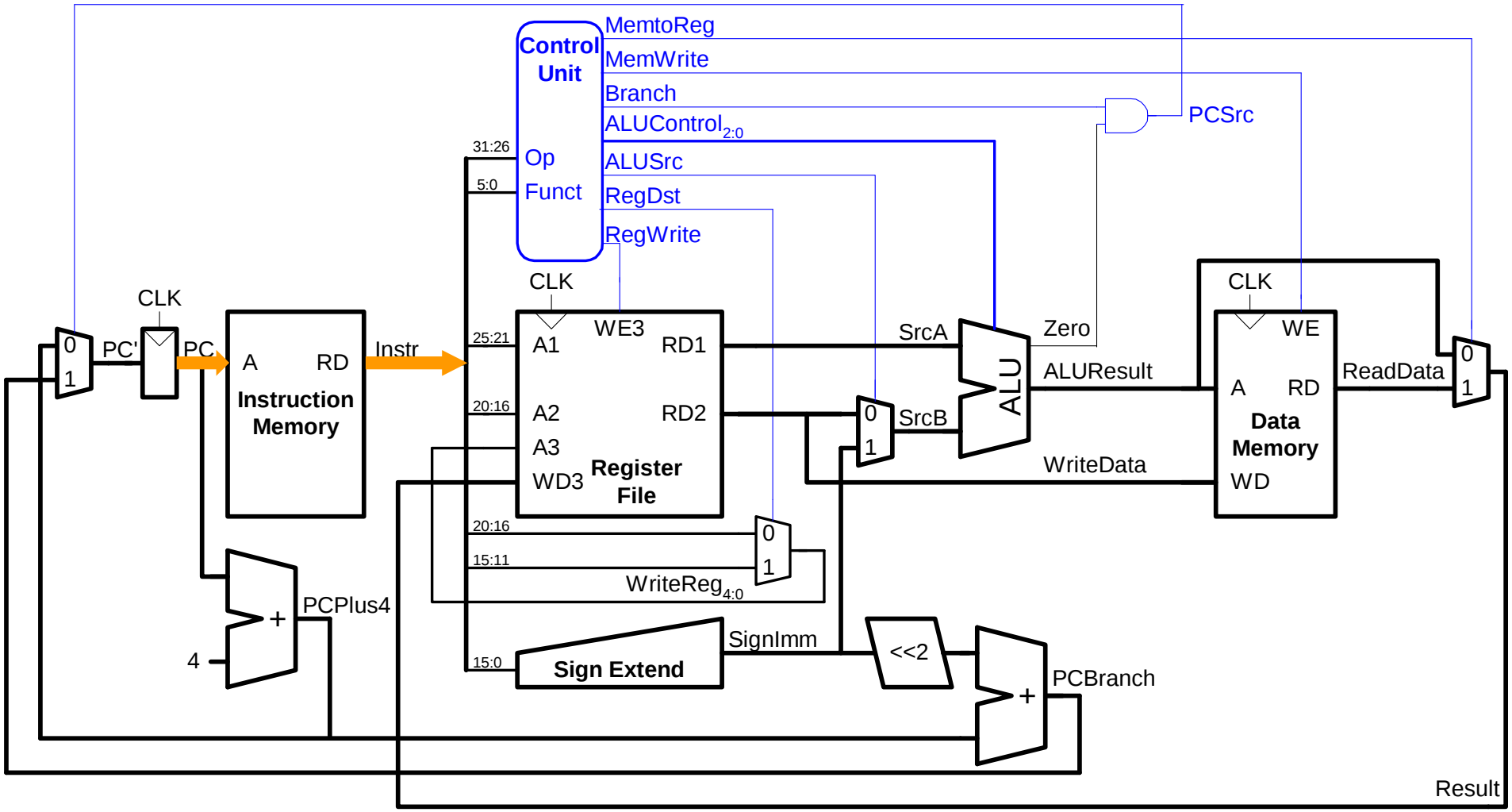
Microarquitectura único ciclo

Store:

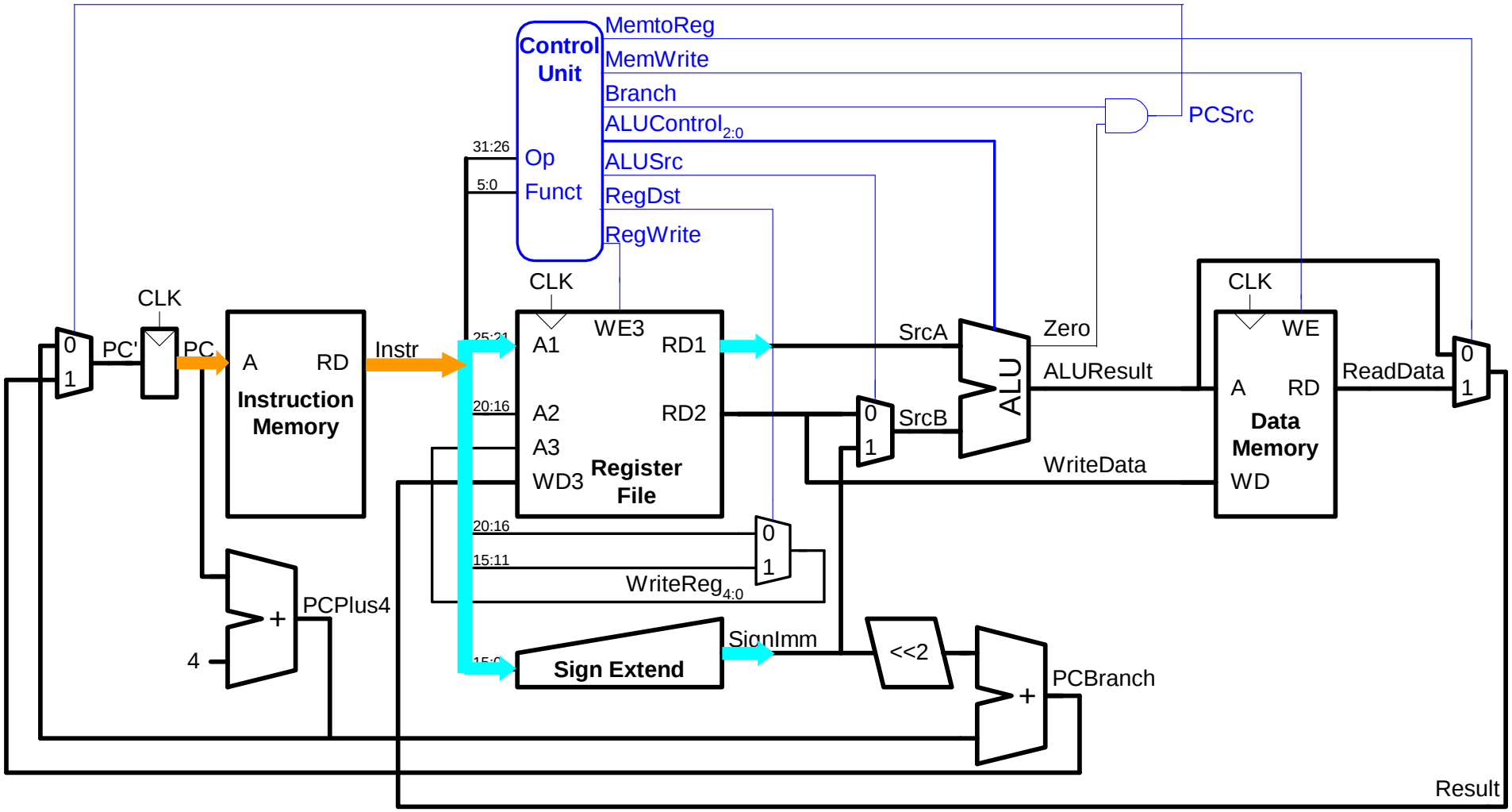
- Mem[d + #valor] ← R[s]
- PC ← PC + 4

1) Traer la instrucción
2) Obtener los operandos: leer los registros *s* y *d* del banco de registros y el valor inmediato.
3) Calcular la dirección destino
4) Escribir el valor del registro en memoria.
5) Actualizar el PC

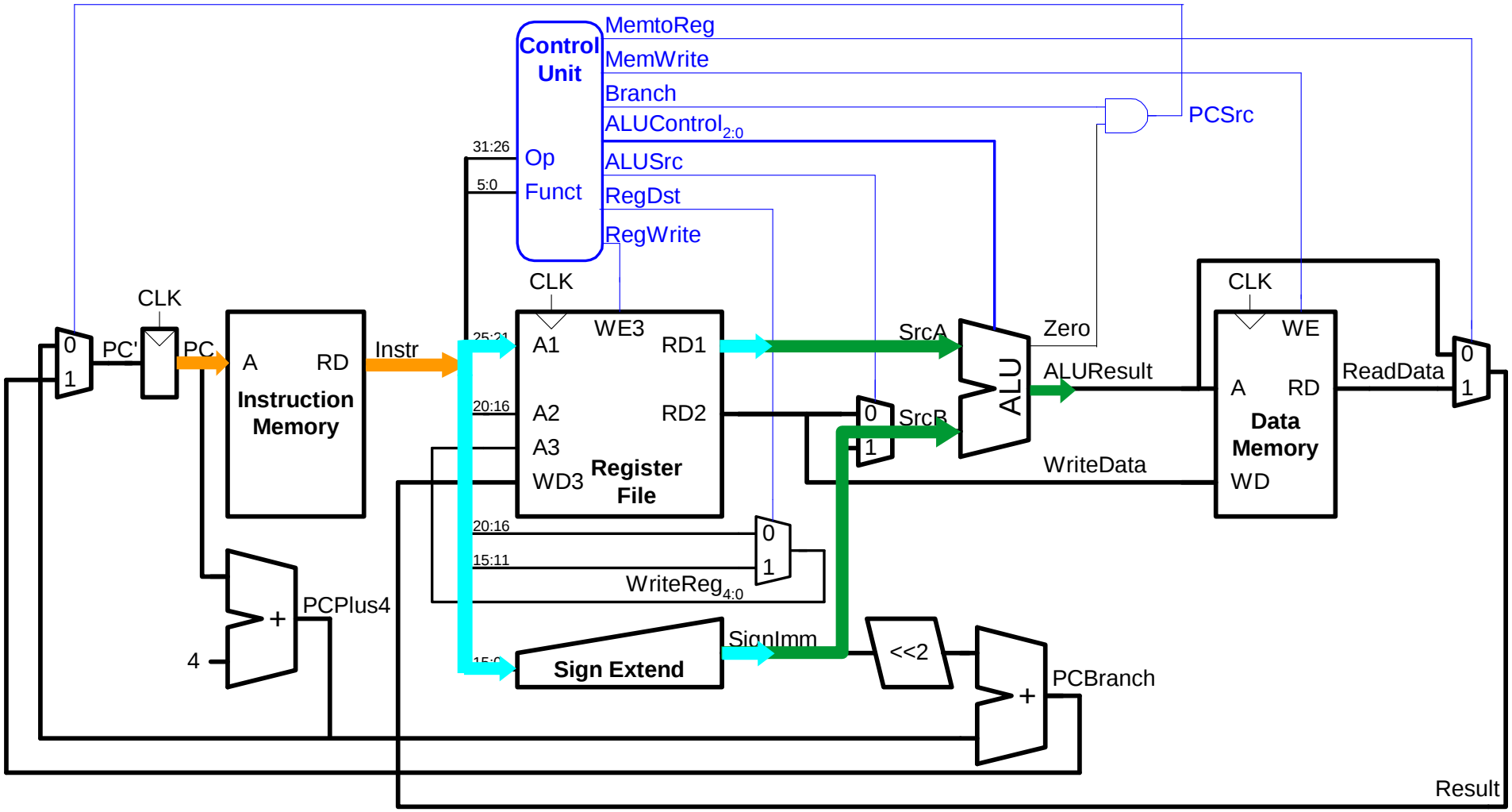
Store: Traer la instrucción



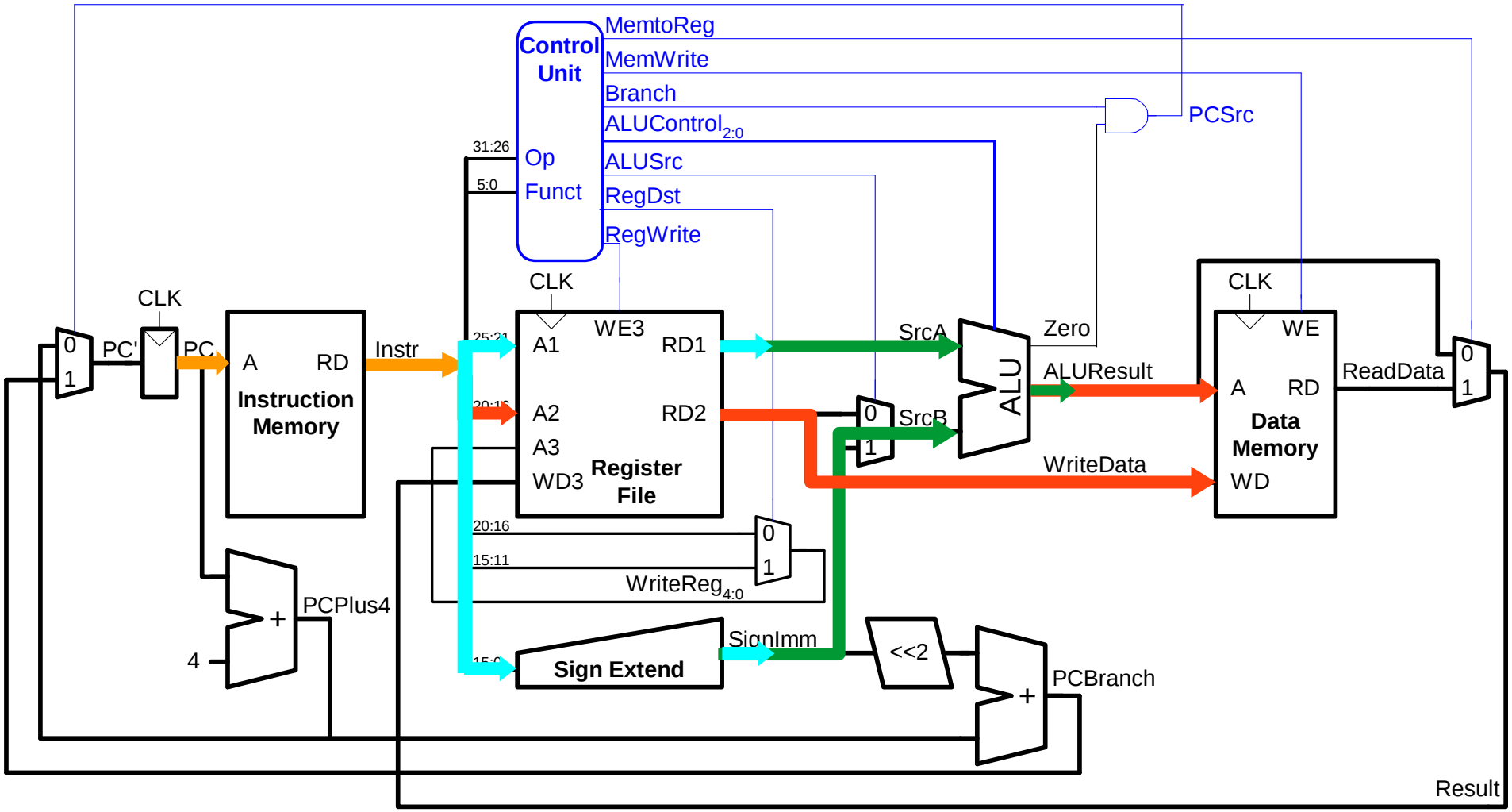
Store: Obtención de operandos



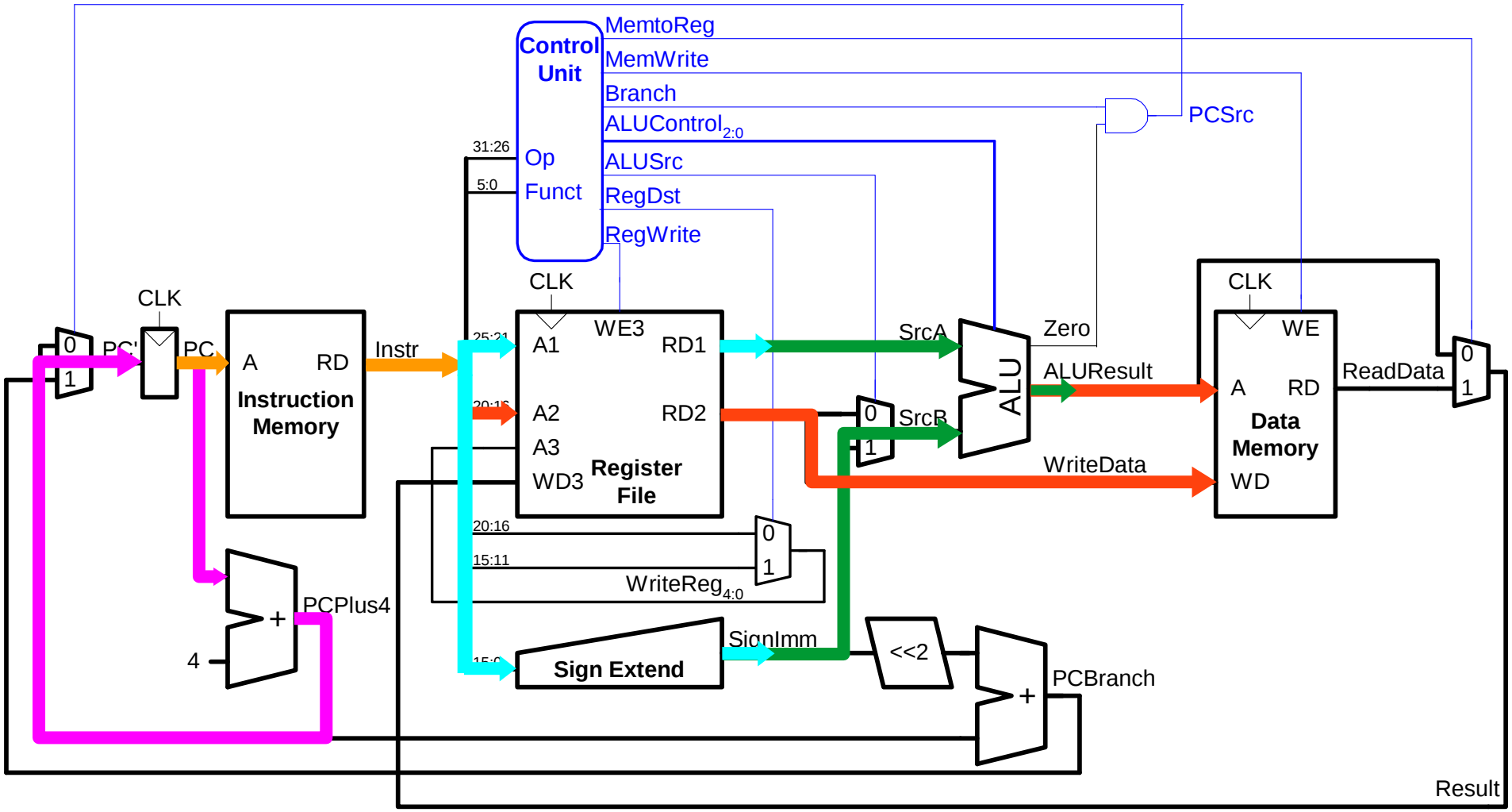
Store: Calcular la dirección



Store: Escribir el resultado



Store: Actualizar PC



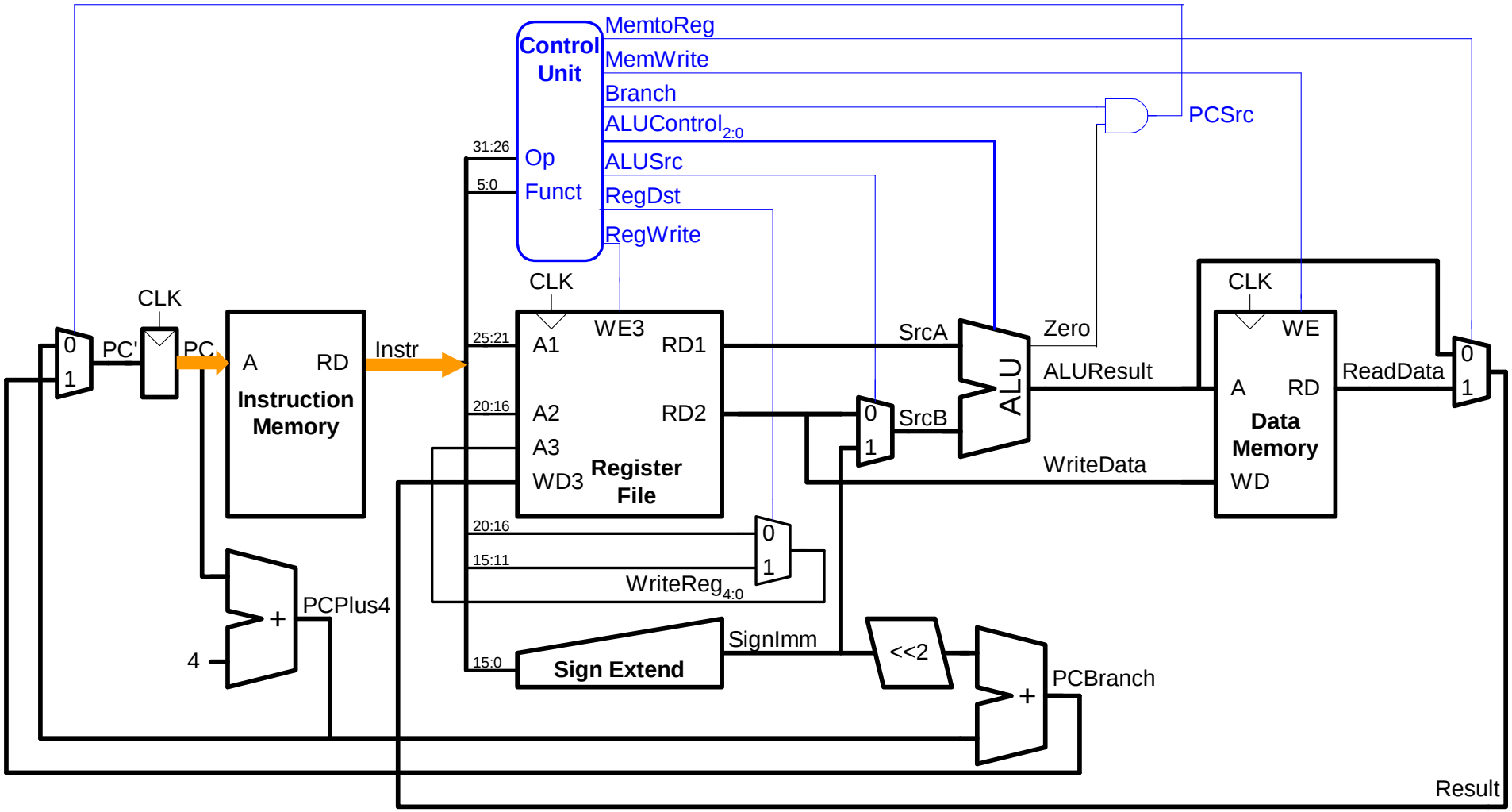
Microarquitectura único ciclo

Branch (beq):

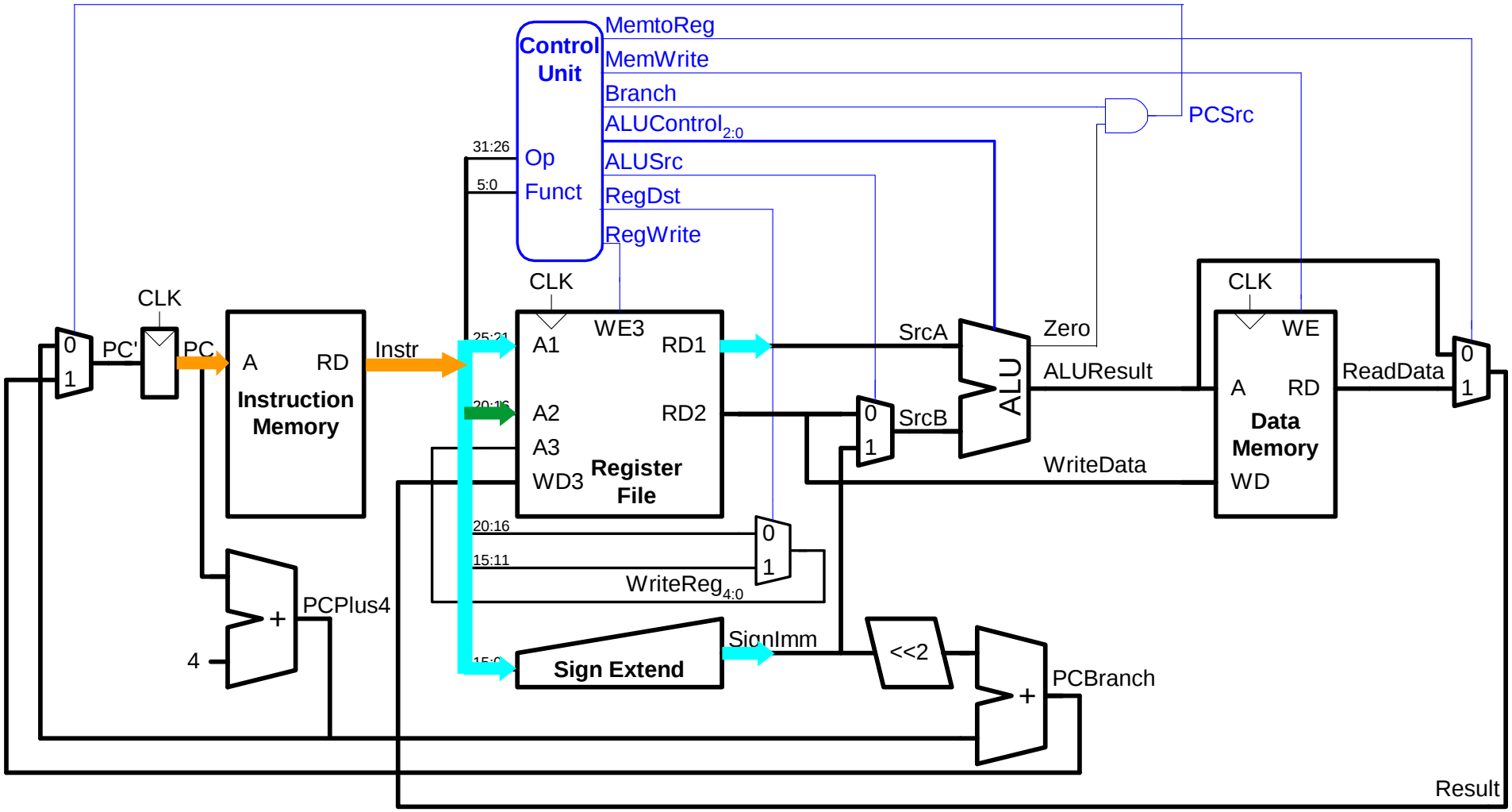
- Si R[s] == R[t]
  - Entonces PC ←PC + offset
  - Si no PC ← PC + 4

1) Traer la instrucción
2) Obtener los operandos: leer los registros *s* y *t* del banco de registros y el valor inmediato.
3) Evaluar la condición
4) Actualizar el PC en función de la condición

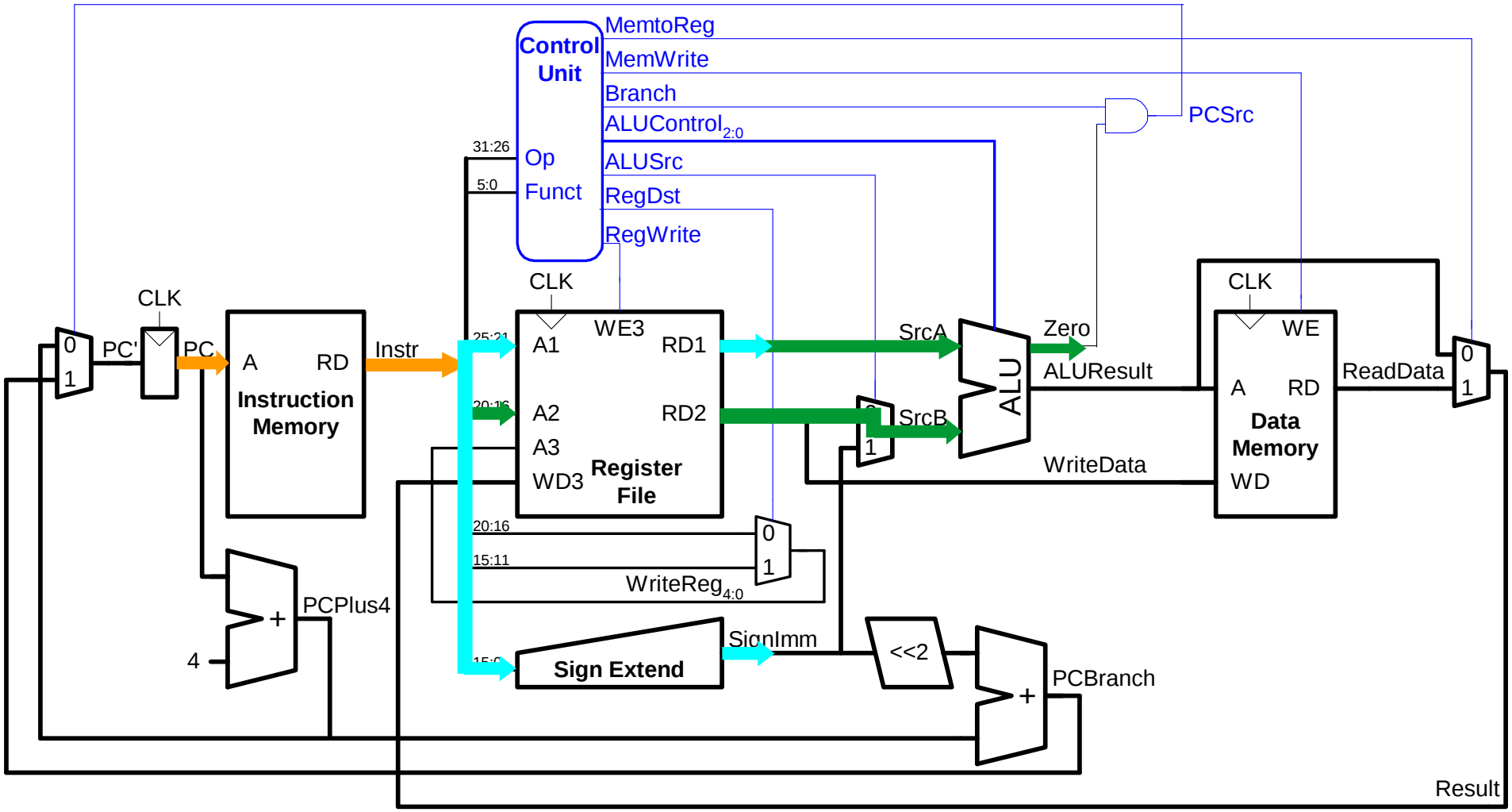
Branch: Traer la instrucción



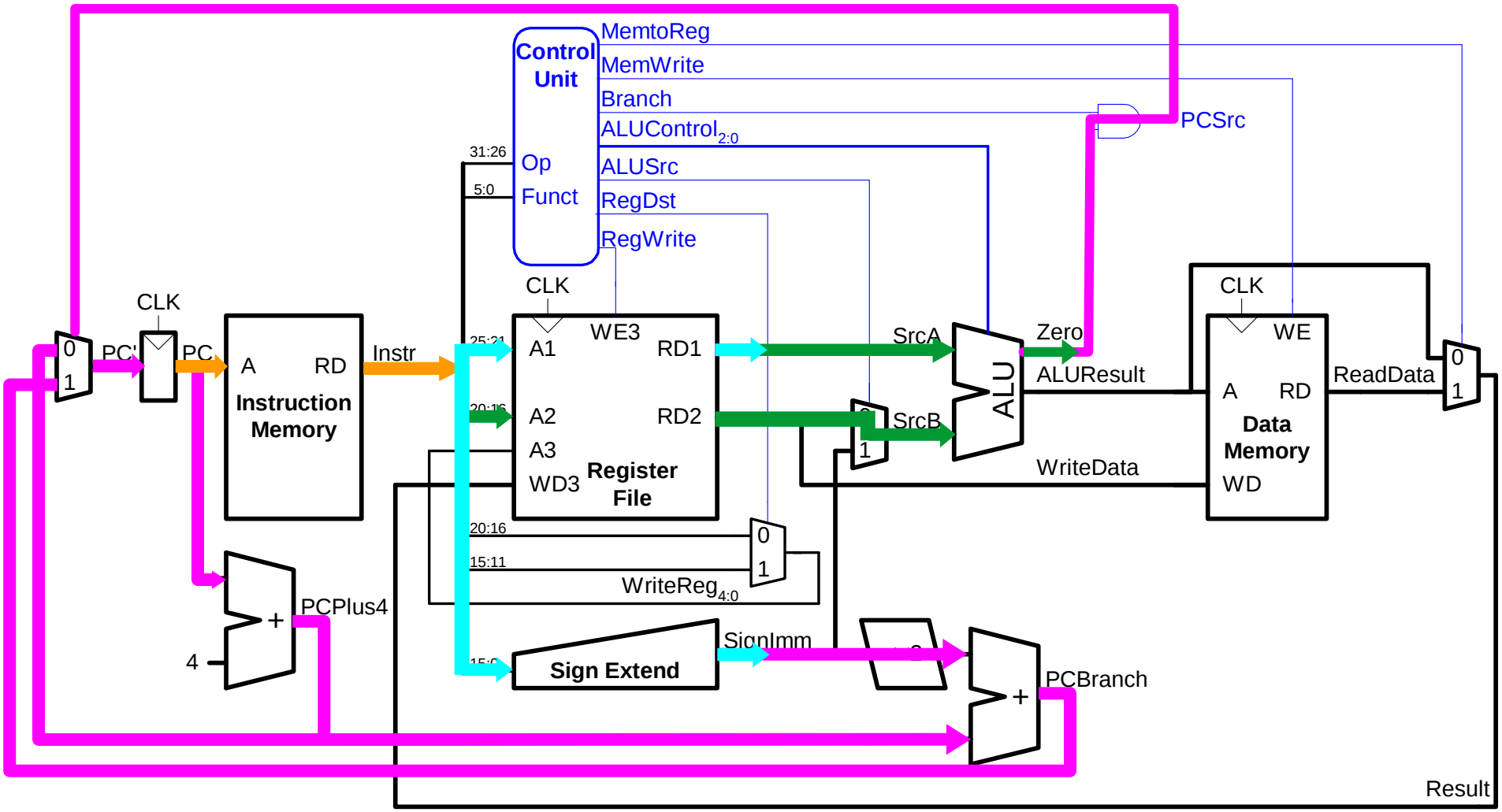
Branch: Obtención de operandos



Branch: Evaluar la condición



Branch: Actualizar PC



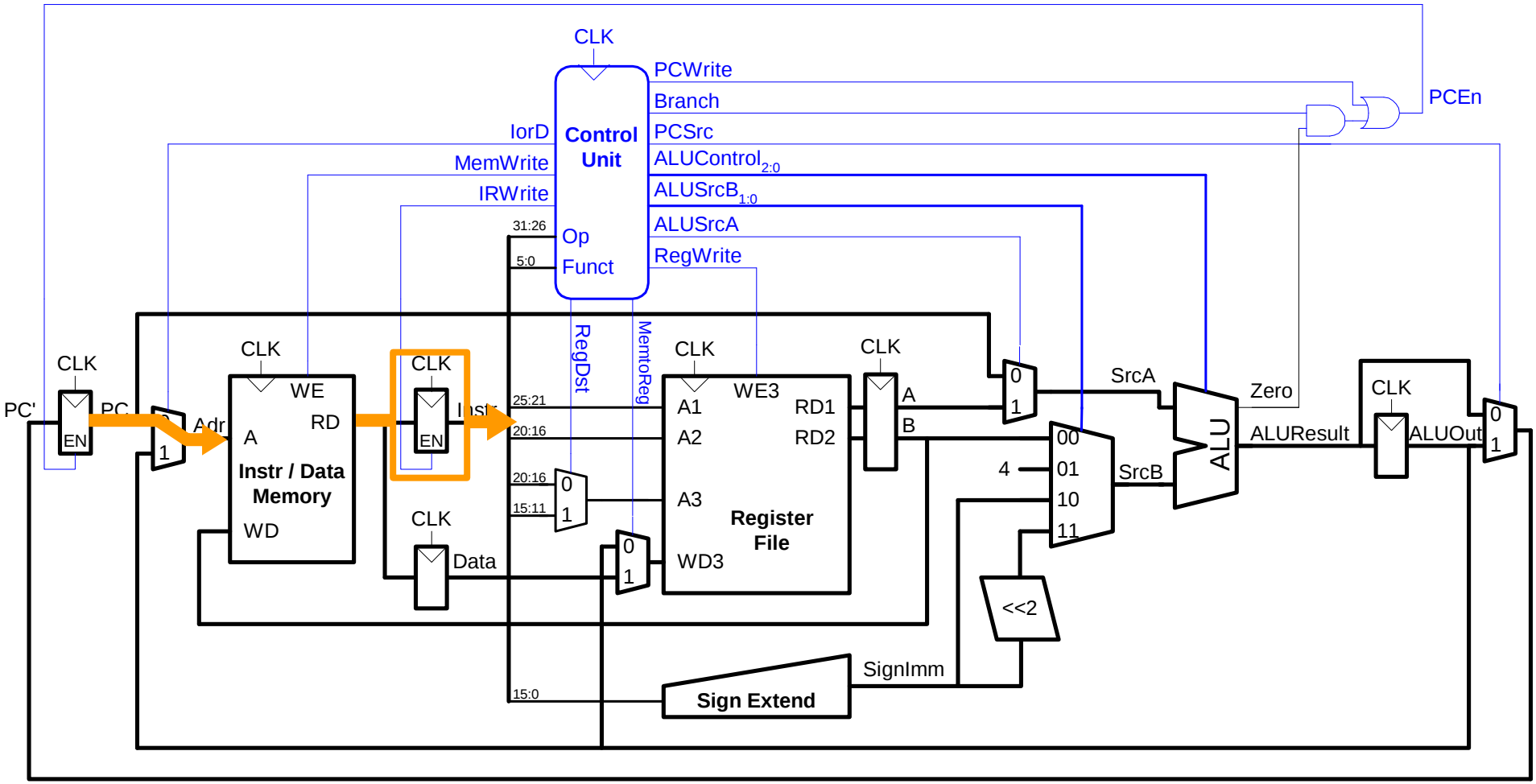
Microarquitectura único ciclo

- Implica:
  - ✔ Diseño simple
  - ✗ El período del ciclo está limitado por la instrucción más lenta.
  - ✗ Necesita 2 sumadores además de la ALU
  - ✗ Fuerza memorias separadas
  - ✗ La operación del datapath es puramente combinacional.
    La máquina está en estados estables solamente al principio y al final del ciclo.

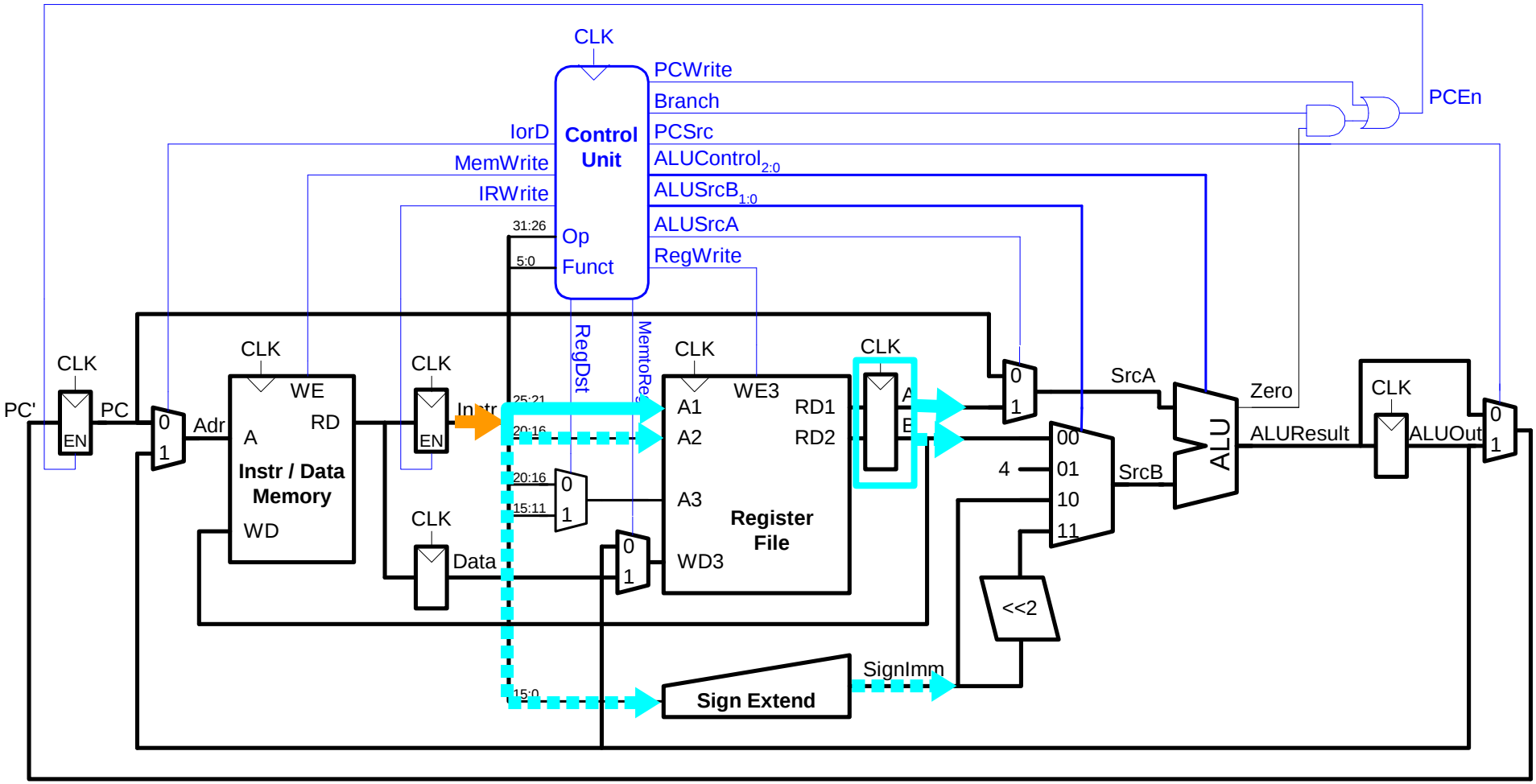
Microarquitectura multi-ciclo

- Las instrucciones se ejecutan a lo largo de varios ciclos cortos.
- Reduce el hardware ya que permite reusar bloques de hardware.
- Agrega hardware para almacenar resultados intermedios.
- Ejecuta una instrucción por vez y cada instrucción demanda múltiples ciclos.

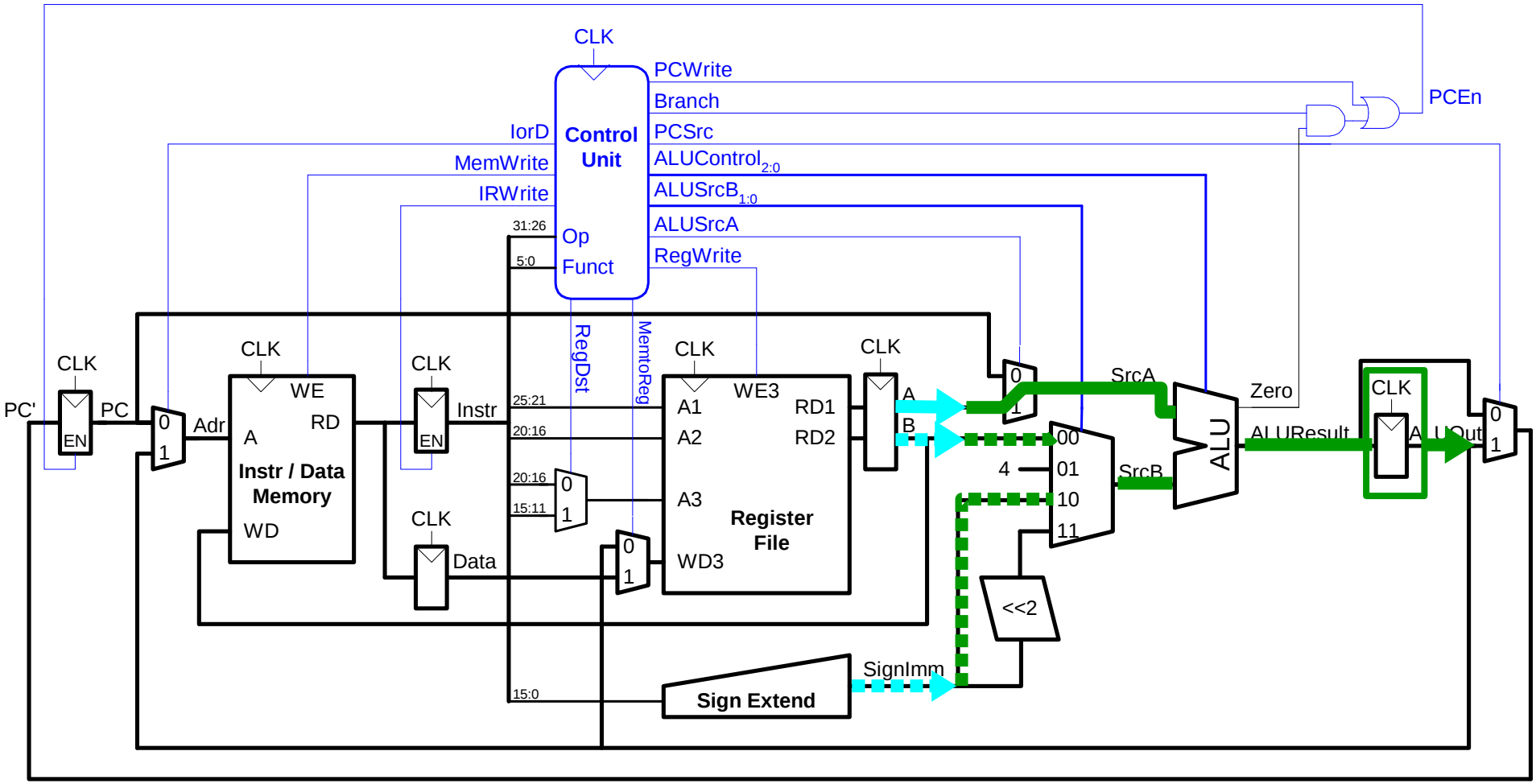
Traer la instrucción



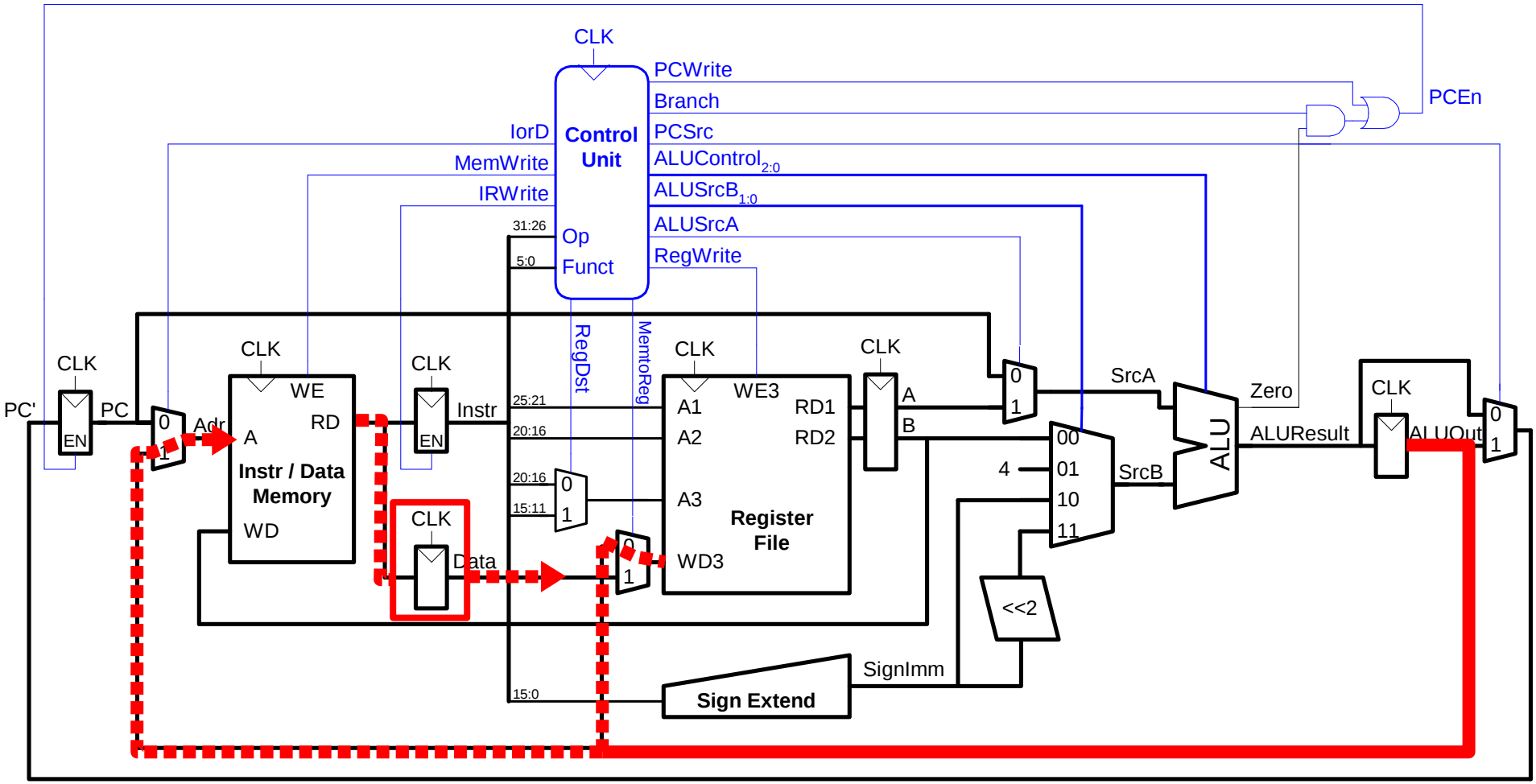
Obtención de operandos



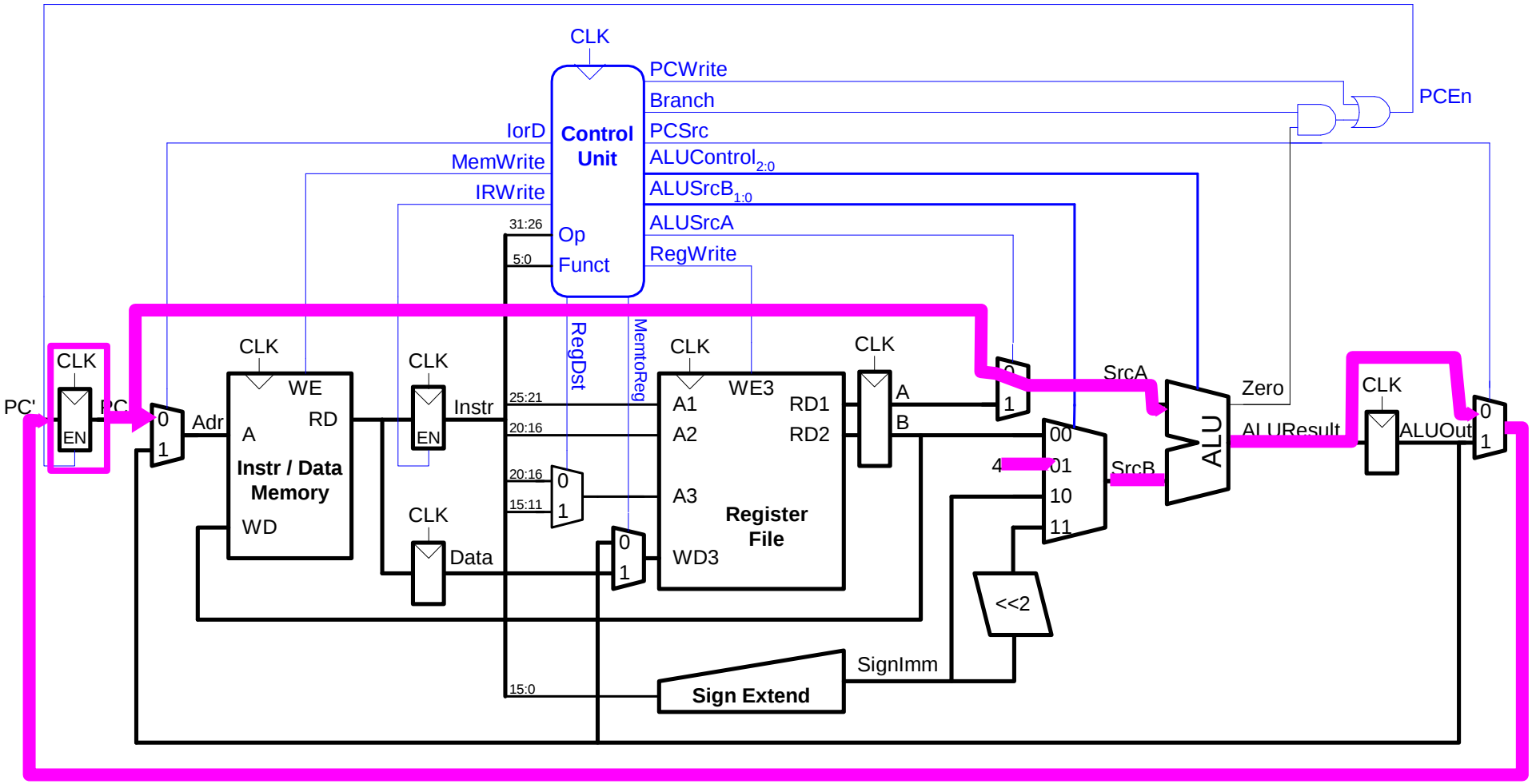
Operación ALU / Cálculo dirección



Escritura en memoria / registros



Actualización del PC



Microarquitectura multi-ciclo

- ✔ Permite mayor frecuencia de reloj.
- ✗ No todos los pasos son de igual duración.
  El reloj debe ser permitir el paso más lento.
- ✗ Agrega la sobrecarga de los registros entre los pasos.

Microarquitectura en pipeline

- Se divide la microarquitectura único ciclo en etapas.
- Cada instrucción se ejecuta en varios ciclos, pero todas en la misma cantidad de ciclos.
- Las etapas se ejecutan en pipeline y diferentes etapas pueden ejecutarse en paralelo.

- Los procesadores modernos están diseñados en pipeline.

Bibliografía

- *Capítulo 1*. Jean-Loup Baer. *Multiprocessor Architecture. From simple pipelines to chip multiprocessor*. Cambridge University Press. 2010.
- *Capítulo 7.* David Money Harris & Sarah L. Harris. *Digital Design and Computer Architecture*. Elsevier. 2013, 2da Ed.
- *Capítulo 4.* David A. Patterson & John L. Hennessy. *Computer Organization and Design. The Hardware/Software Interface.* Elsevier Inc. 2014, 5ta Ed.