Sección de procesamiento: El camino de datos

Montse Bóo Cepeda

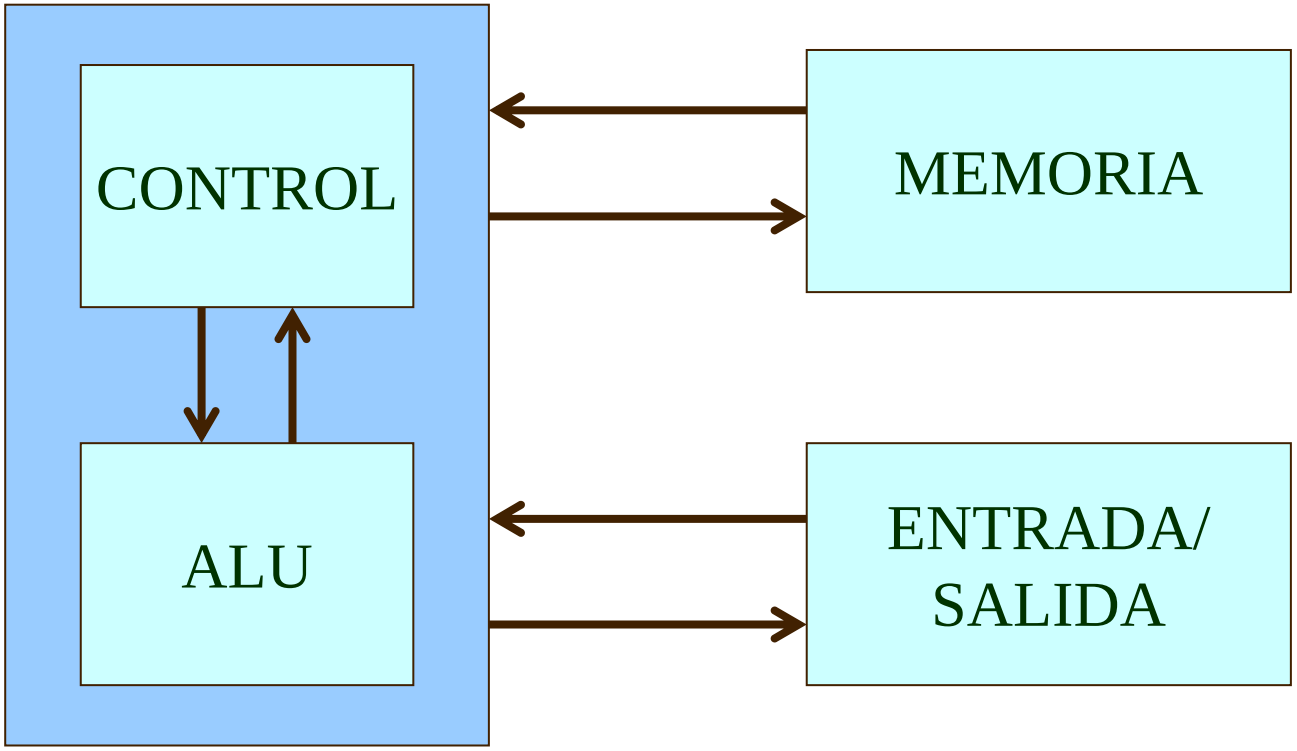
Estructura del curso

1. Evolución y caracterización de los computadores.
2. Arquitectura del MIPS: Introducción.
3. Tipo de datos.
4. El repertorio de instrucciones.
5. Aritmética del computador.
6. El camino de datos.
7. Sección de control.
8. El camino de datos multiciclo.
9. Sección de control multiciclo.
10. Entrada/Salida (I/O).

Esquema de contenidos

1. Estructura básica del procesador
2. El ciclo de instrucción
3. Diseño de un procesador sencillo

Estructura básica de un procesador



Arquitectura Von Neumann

Estructura básica de un procesador

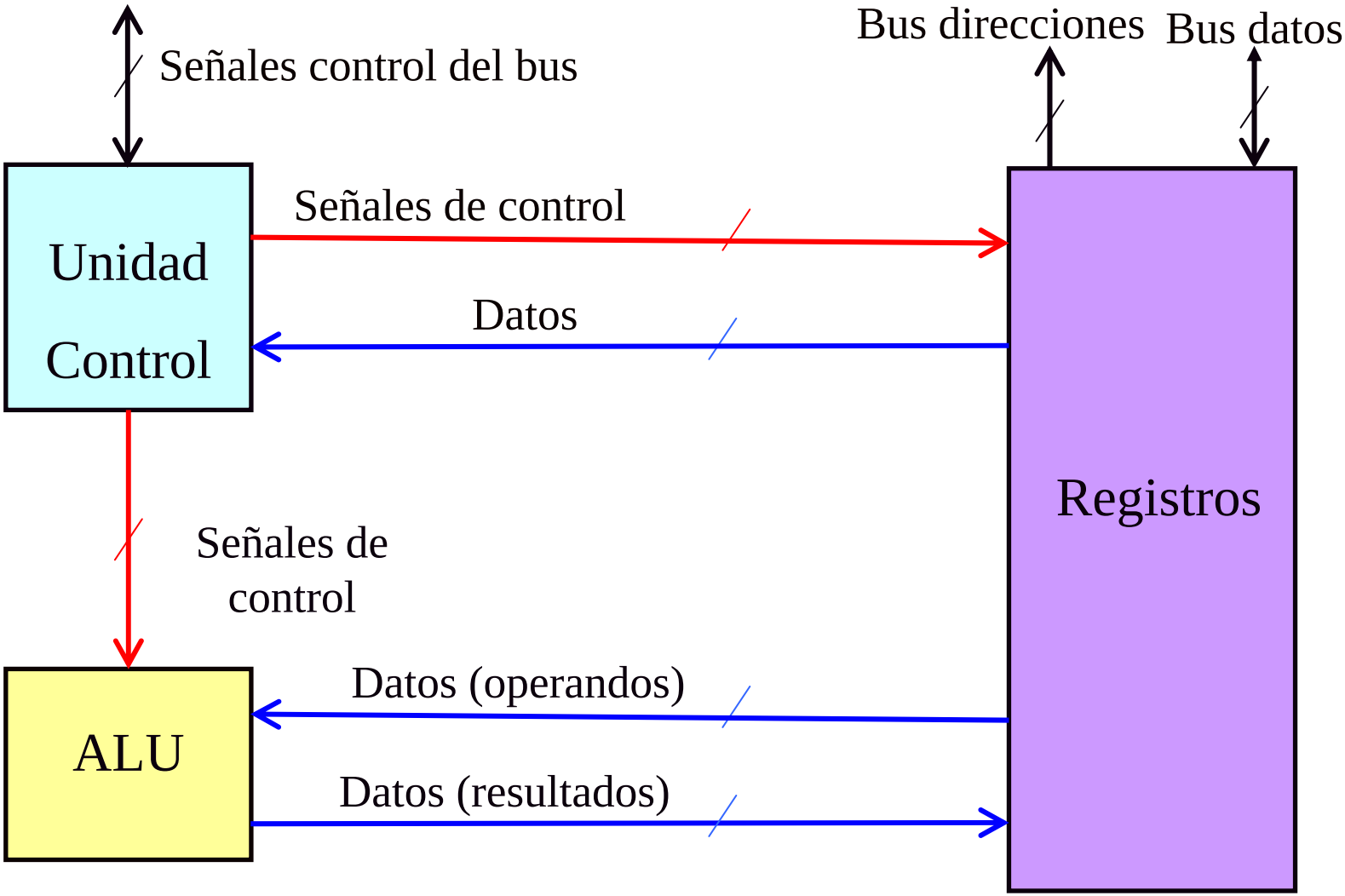
- El procesador es el que se encarga de ejecutar las instrucciones especificadas por el programa.
- Funciones básicas:
  - **Captar instrucciones**. El procesador debe leer instrucciones de la memoria
  - **Interpretar instrucciones**. La instrucción debe decodificarse para determinar qué acción es necesaria
  - **Captar datos**. La ejecución puede exigir leer datos de la memoria o de un módulo de E/S
  - **Procesar datos**. La ejecución de una instrucción puede exigir llevar a cabo alguna operación aritmética o lógica
  - **Escribir datos**. Los resultados de una ejecución pueden tener que ser escritos en la memoria o en un módulo de E/S

Estructura básica de un procesador

El procesador se compone de varias partes:

1. Unidad de control
2. Unidad aritmético-lógica
3. Un banco de registros
4. Otros registros internos:
   - El contador de programa (PC)
   - Registro de instrucciones (IR)
5. Buses, multiplexores, memorias…

Estructura básica de un procesador

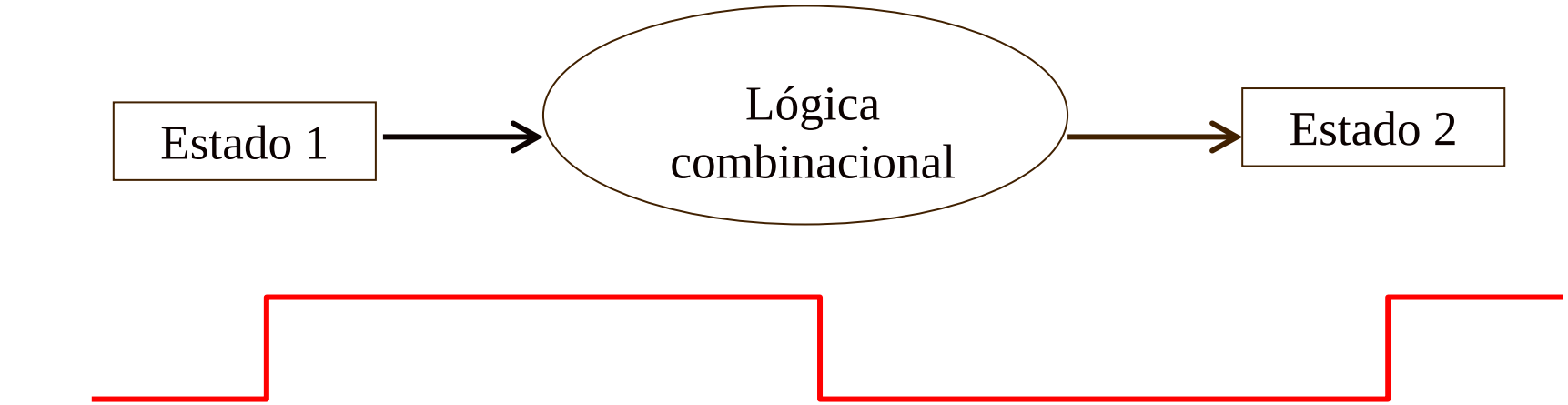


Metodología de sincronización

- El camino de datos se basa en transferencias/computaciones entre registros:
  - Elementos combinacionales:
    - Su salida depende únicamente de las entradas
    - Ejemplo: ALU (sumador, multiplexor, desplazador,…)
  - Elementos secuenciales:
    - Operan sobre el estado del sistema.
    - La salida depende de la entrada y del estado.
    - Ejemplo: Memorias y registros
  - La metodología de sincronización define cuándo pueden leerse y escribirse la diferentes señales
  - Asumimos sincronización por flancos

Metodología de sincronización

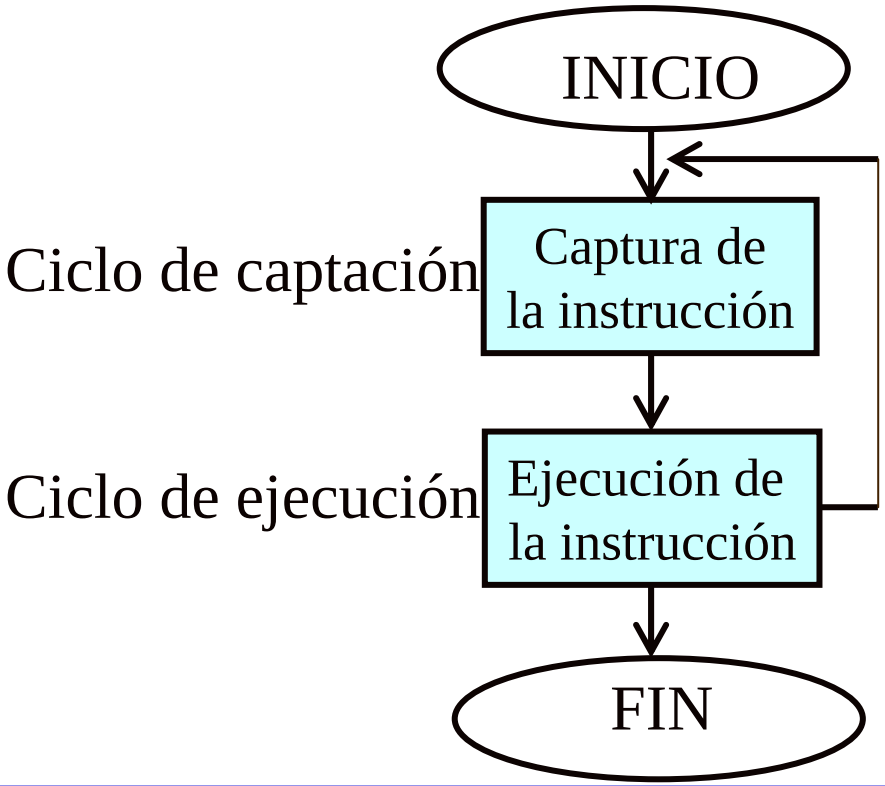
Estrategia de sincronización por flancos



Ciclo de reloj

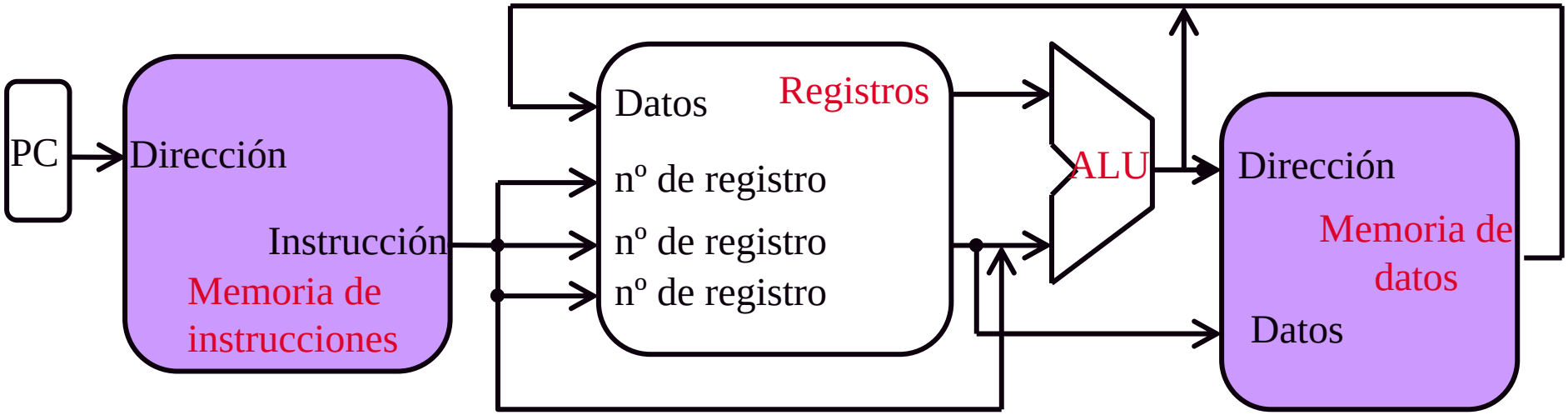
El ciclo de instrucción

- El procesamiento que requiere una instrucción se denomina ciclo de instrucción.
- Ciclo básico de instrucción:



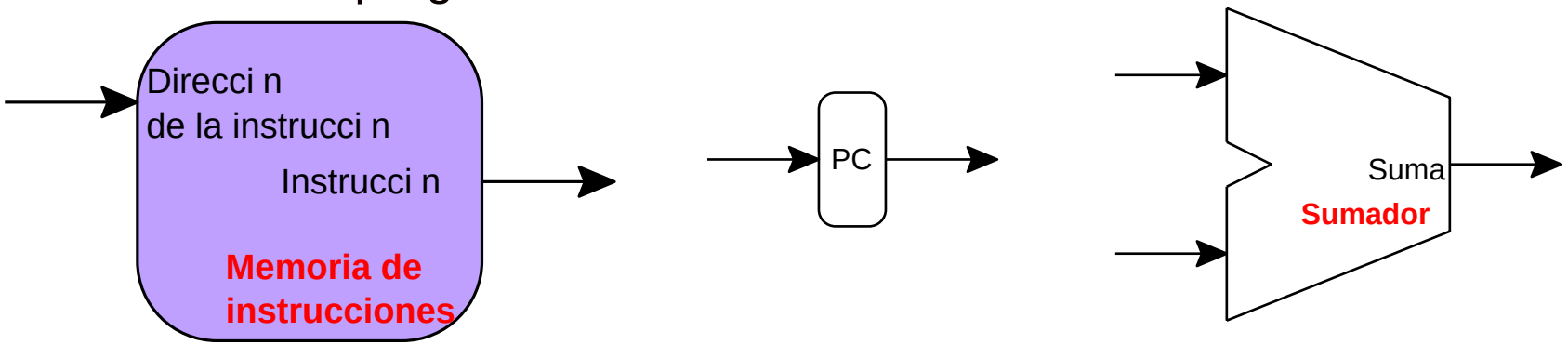
Diseño del camino de datos

- Vamos a construir un camino de datos y su unidad de control para dos realizaciones diferentes de un subconjunto del repertorio de instrucciones del MIPS:
  - Instrucciones de acceso a memoria: lw, sw
  - Instrucciones aritmético-lógicas: add, sub, or, slt
  - Instrucción de salto condicional: beq
  - Instrucción de salto incondicional: j

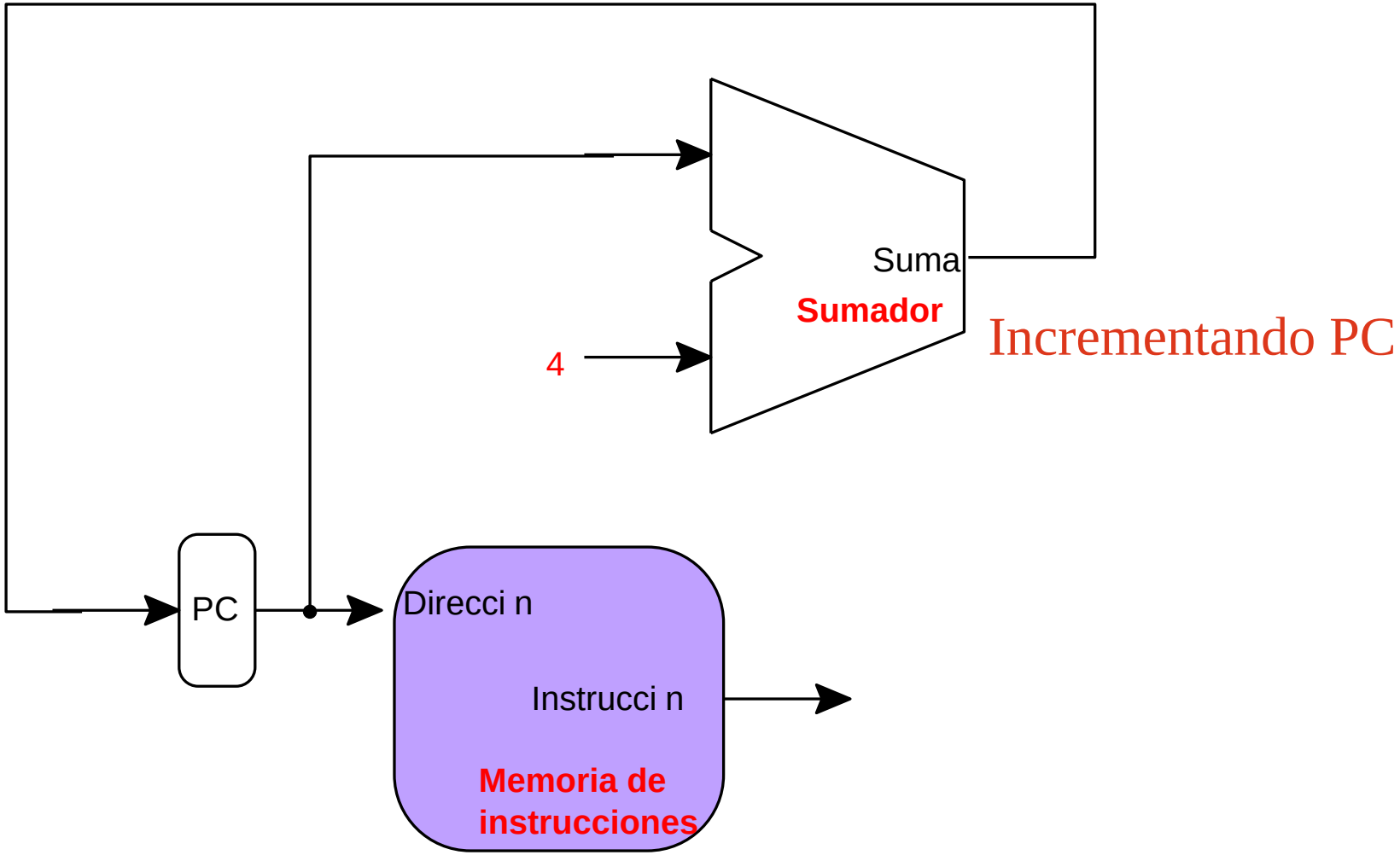


Contador de programa

- El camino de datos implementa el ciclo completo de instrucción: “carga-decodificación-ejecución”.
- Cada instrucción está en una posición de memoria específica.
- Contador de programa:
  - Almacenamos la dirección de la instrucción en el PC.
  - Tras procesar una instrucción, avanzamos el contador hasta la siguiente instrucción.
- Elementos para la búsqueda de instrucciones y el incremento del contador de programa:



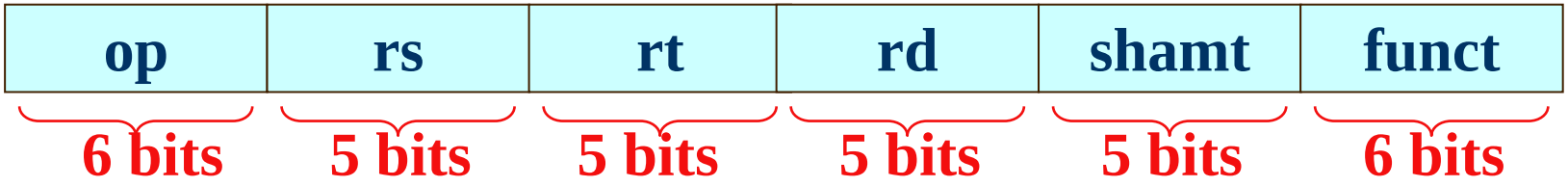
Contador de programa



Instrucciones aritmético y lógicas

TIPO R: Operaciones aritméticas y lógicas

TIPO R

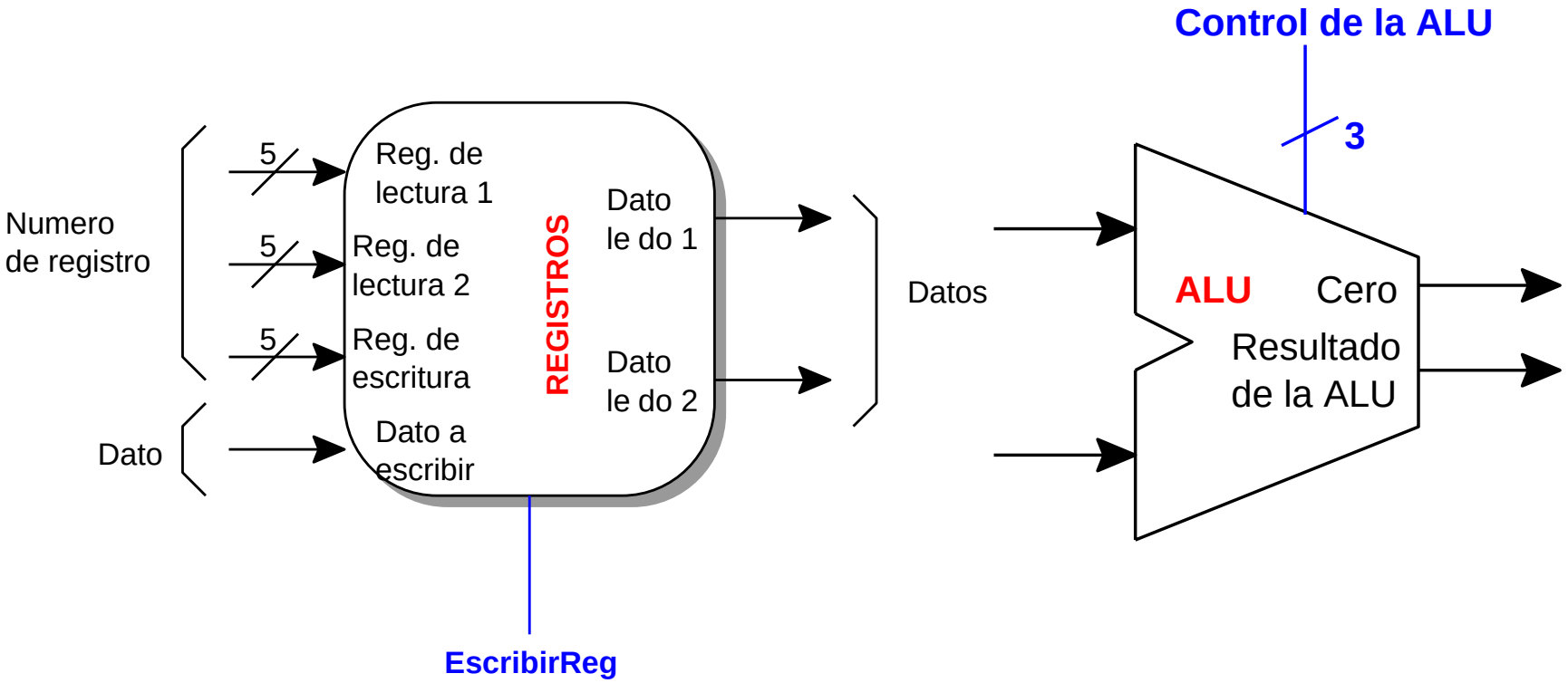


Campos:

- op: código de operación.
- rs: primer registro operando fuente
- rt: segundo registro operando fuente
- rd: registro operando destino
- shamt: tamaño de desplazamiento (shift amount)
- funct: código de función.

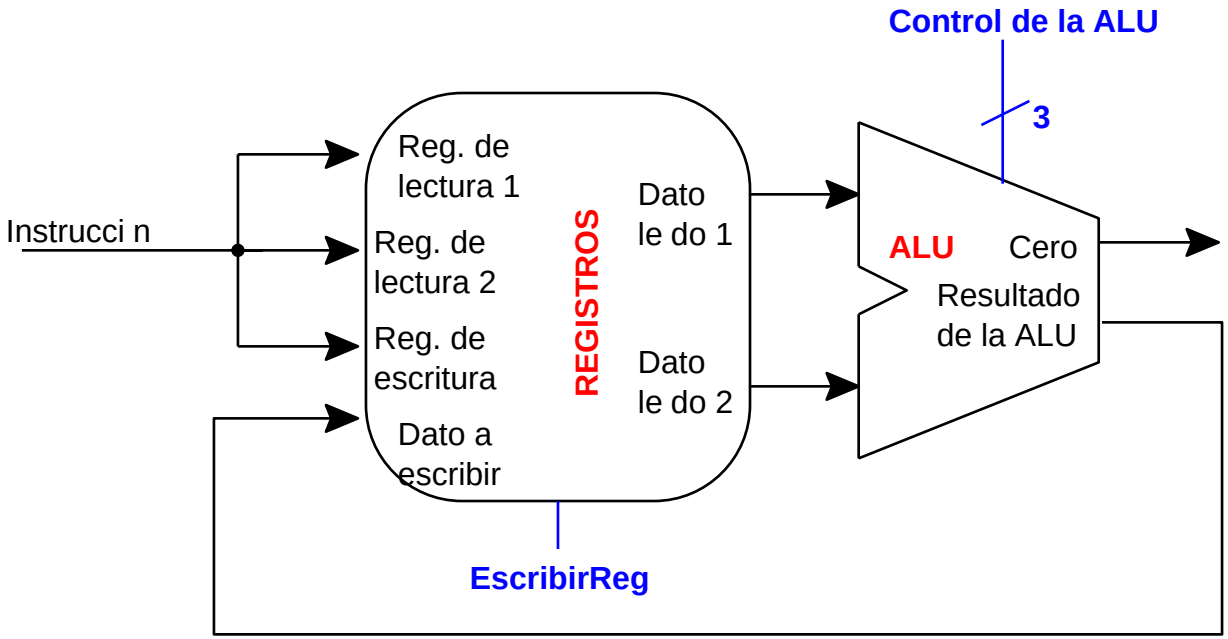
Instrucciones aritmético y lógicas

- Elementos básicos para la realización de instrucciones tipo R



Instrucciones aritmético y lógicas

- Operaciones tipo R:
  - Involucran tres registros: dos de lectura y uno de escritura.
  - Usan la ALU para realizar operaciones

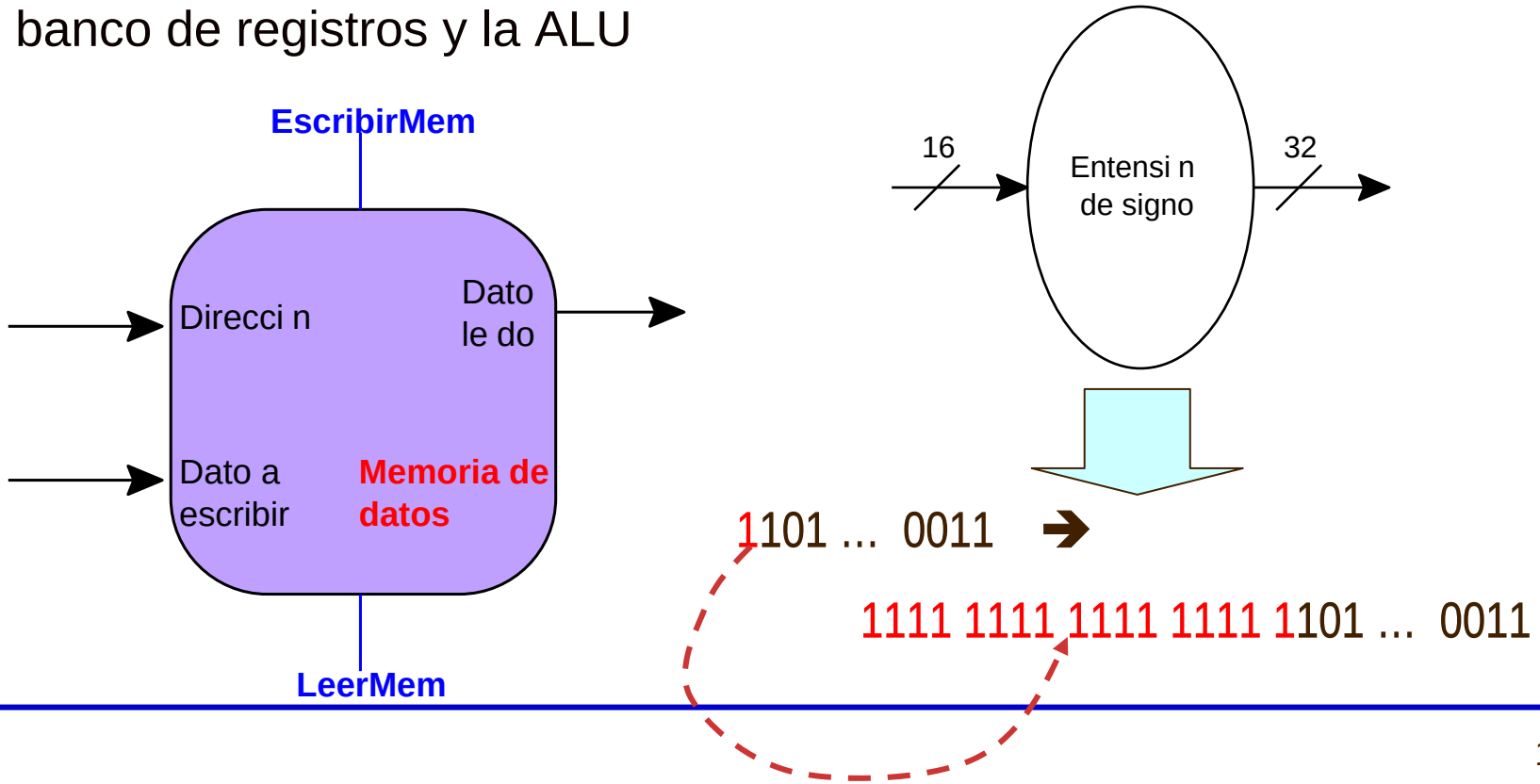


Instrucciones de carga y almacenamiento

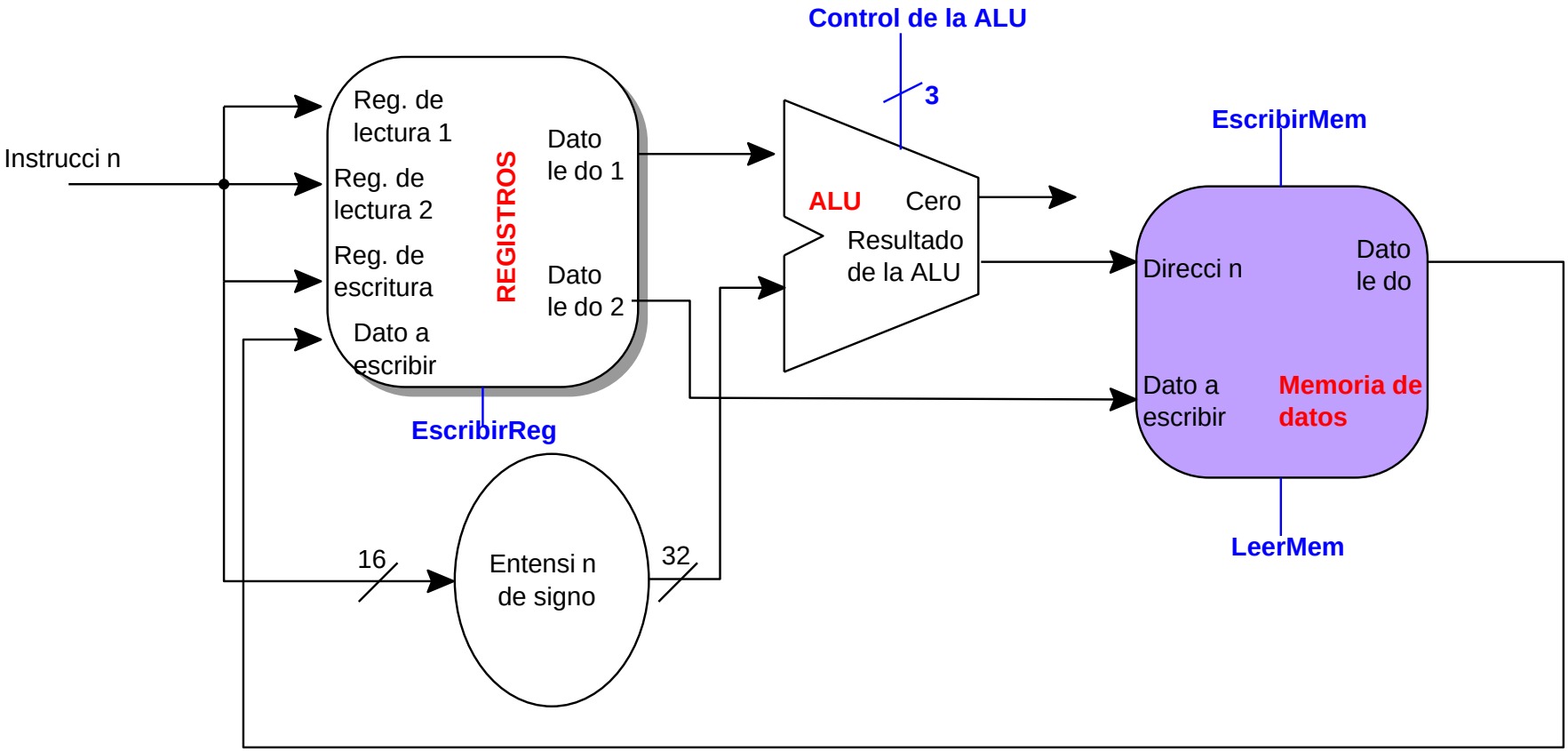
- **Ejemplo: lw $t1, offset($t2)**
  - Dirección almacenada en $t2 + offset
  - **lw**: Lectura memoria, escritura en registro $t1
- Computación de la dirección:
  - Extensión del signo del offset (16-bits) a una palabra de 32 bits con signo
- *Requerido hardware para la extensión de signo*

Instrucciones de carga y almacenamiento

- Load: realiza un acceso al banco de registros, calcula la dirección de memoria, lee de memoria y escribe en el banco de registros
- Store: realiza un acceso al banco de registros, calcula la dirección de memoria, lee del banco de registros y escribe en memoria.
- Elementos para la realización de loads y stores, además del banco de registros y la ALU



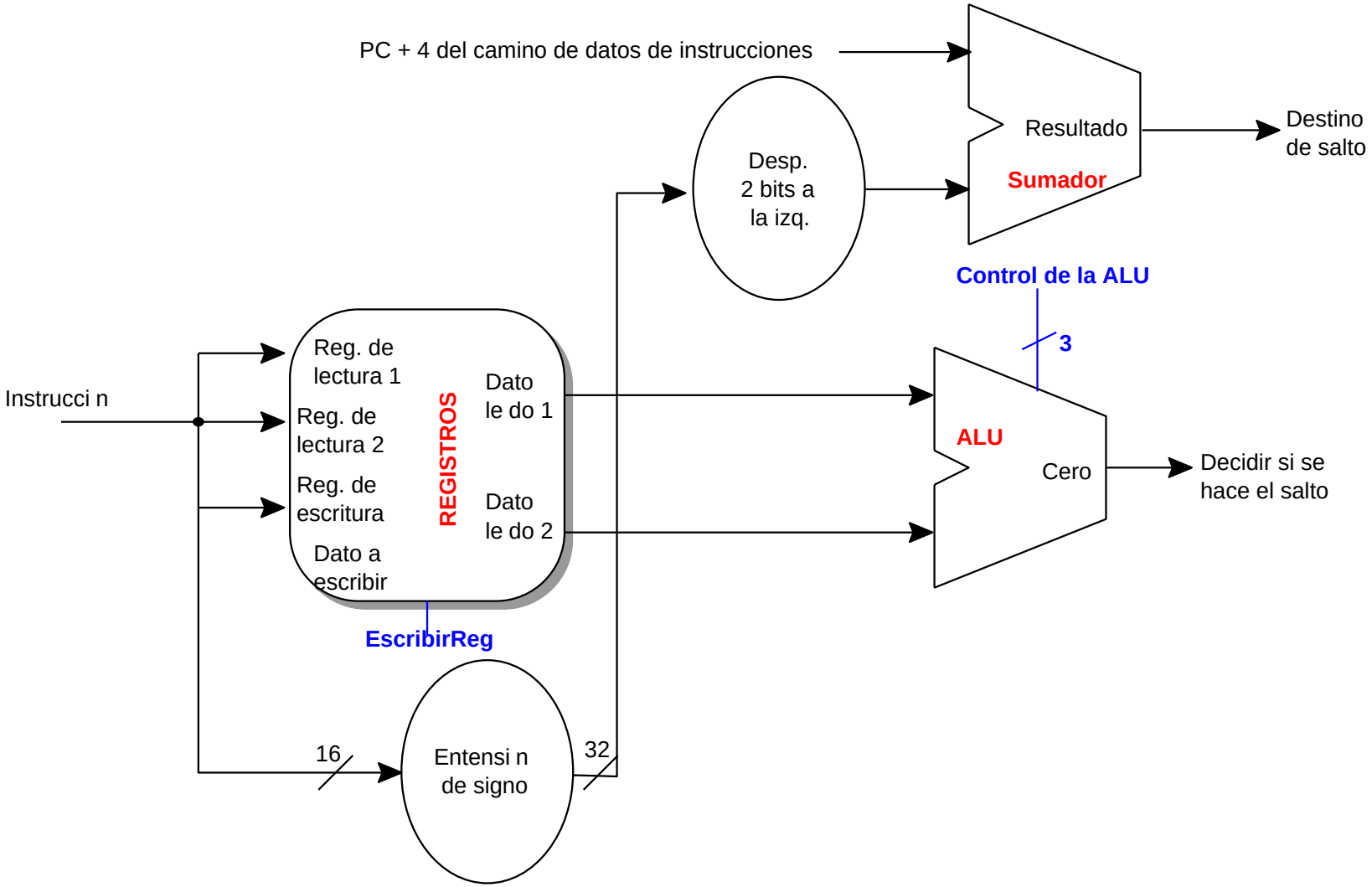
Instrucciones de carga y almacenamiento



Salto condicional

- **Ejemplo: beq $t1, $t2, offset**
  - Se evalúa si dos registros contienen el mismo valor ($t1, $t2)
- El salto es relativo. El offset es con signo y de 16 bits y tiene que realizarse una extensión de signo.
- Dirección de salto:
  - La dirección base es la siguiente a la instrucción del salto (PC+4)
  - Hacer un desplazamiento de 2 posiciones a la izquierda del offset (sólo se direccionan palabras completas)
  - Suma el offset desplazado, con extensión de signo, al PC incrementado (PC+4)

Salto condicional

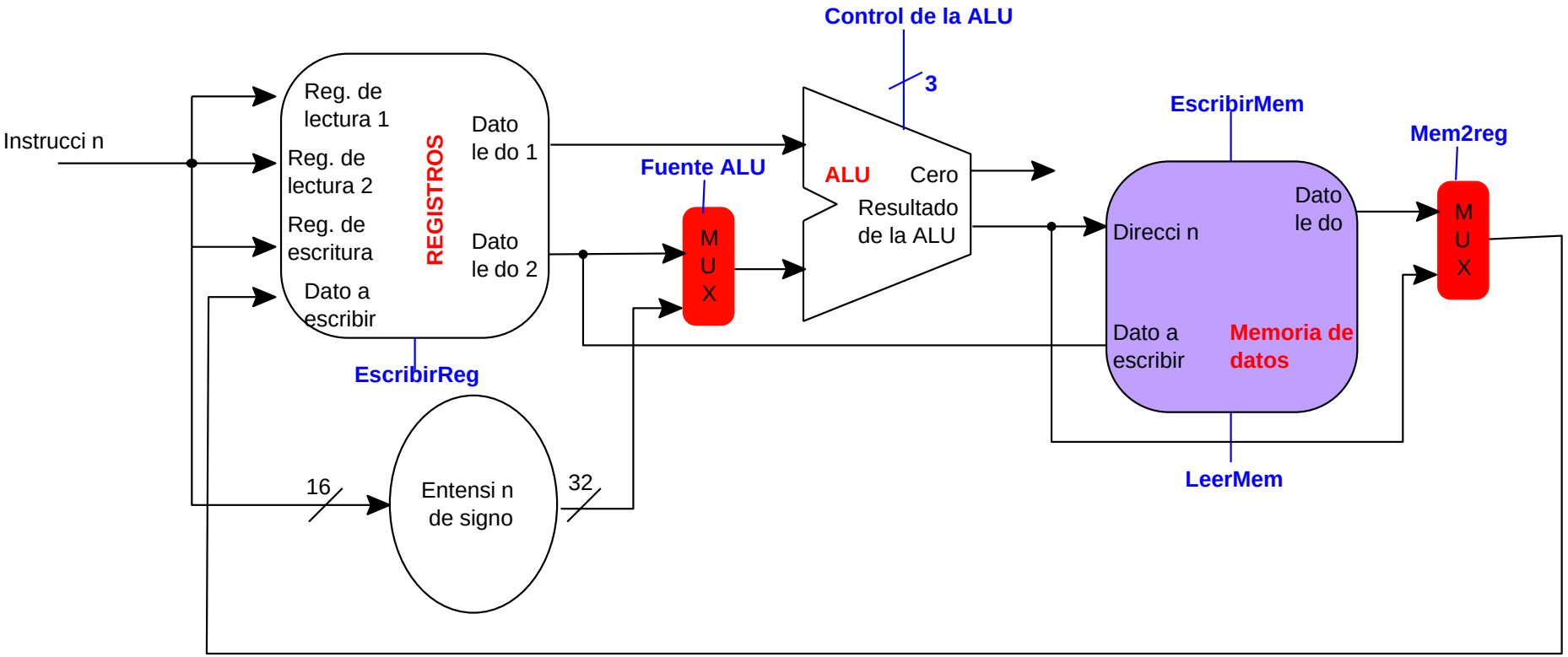


Diseño del camino de datos

- Para construir el camino de datos hemos de combinar los elementos explicados anteriormente que permiten implementar:
  - Instrucciones de acceso a memoria: lw, sw
  - Instrucciones aritmético-lógicas: add, sub, or, slt
  - Instrucción de salto condicional: beq
  - Instrucción de salto incondicional: j
- Reutilización del hardware: hardware compartido selecciona los datos mediante multiplexores
- Parte del hardware no se podrá reutilizar y habrá que replicarlo

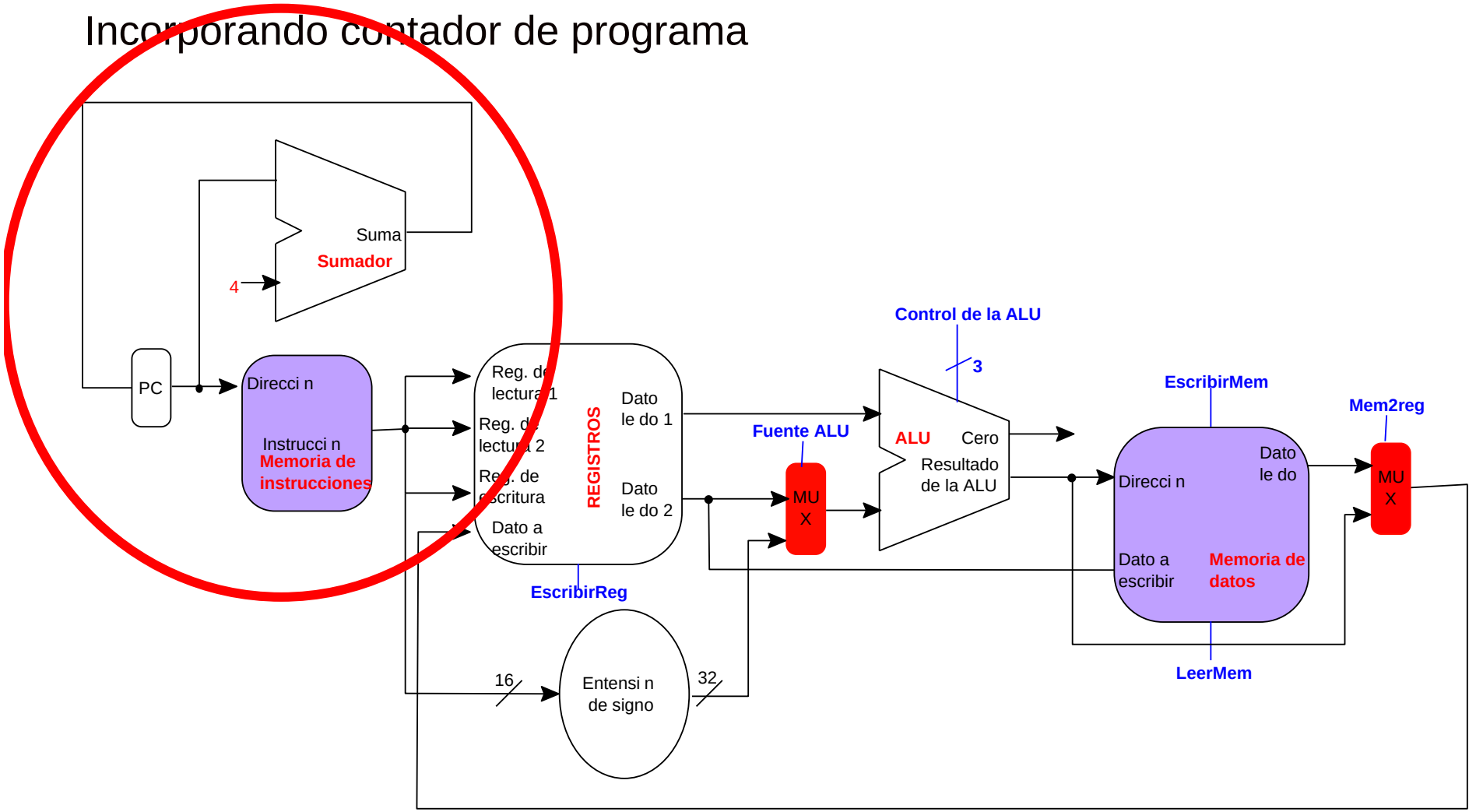
Diseño del camino de datos

Camino de datos para instrucciones de memoria y aritmético-lógicas



Diseño del camino de datos

Incorporando contador de programa



Diseño del camino de datos

Añadimos saltos condicionales