Procesador: camino de datos y unidad de control

Índice

- √ Introducción
- √ Estructura básica de la CPU: camino de datos y control
- √ El ciclo de instrucción
- √ Diseño de un procesador monociclo
- √ Descomposición de la ejecución en etapas
- √ Realización multiciclo
- √ Diseño de la unidad de control
- √ Procesamiento de excepciones

Introducción

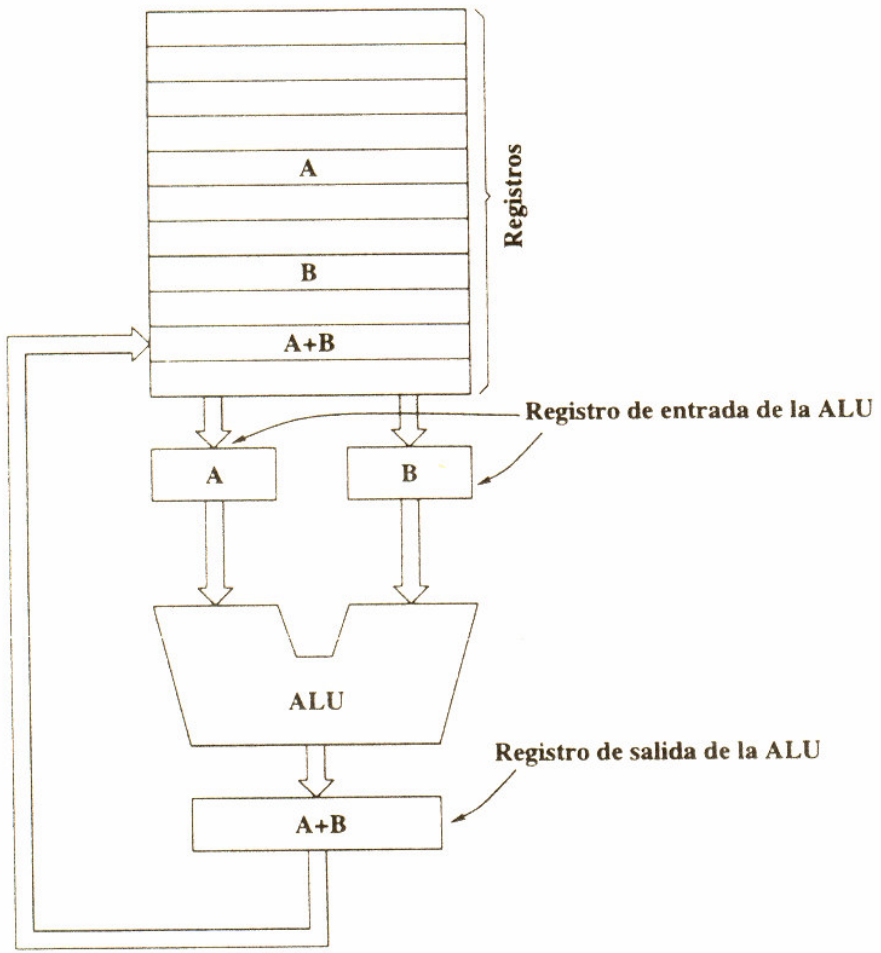
- En este tema se estudia la Unidad Central de Procesamiento (CPU)
  - √ Camino de datos:
    - Es la sección del computador encargada de manipular y transformar los datos procedentes de la memoria o los registros internos, para obtener los resultados
    - Su labor es soportar el conjunto de operaciones que precisan las instrucciones del repertorio que es capas de interpretar la unidad de control
  - √ Unidad de control:
    - Es la sección del computador encargada de interpretar las instrucciones del programa y gobernar la ejecución de las mismas
- La organización de los procesadores ha evolucionado con el paso de los años, guiada por el desarrollo tecnológico y la necesidad de obtener altas prestaciones
- En este tema se analizan las ideas básicas comunes a todos los procesadores, que sientan las bases para poder comprender los avances en arquitectura de computadores

Introducción

- 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. La unidad de control
  2. La unidad aritmético-lógica
  3. Un banco de registros
  4. Otros registros internos, algunos de los más importantes:
     1. El *contador de programa* (PC)
     2. El *registro de instrucciones* (IR)

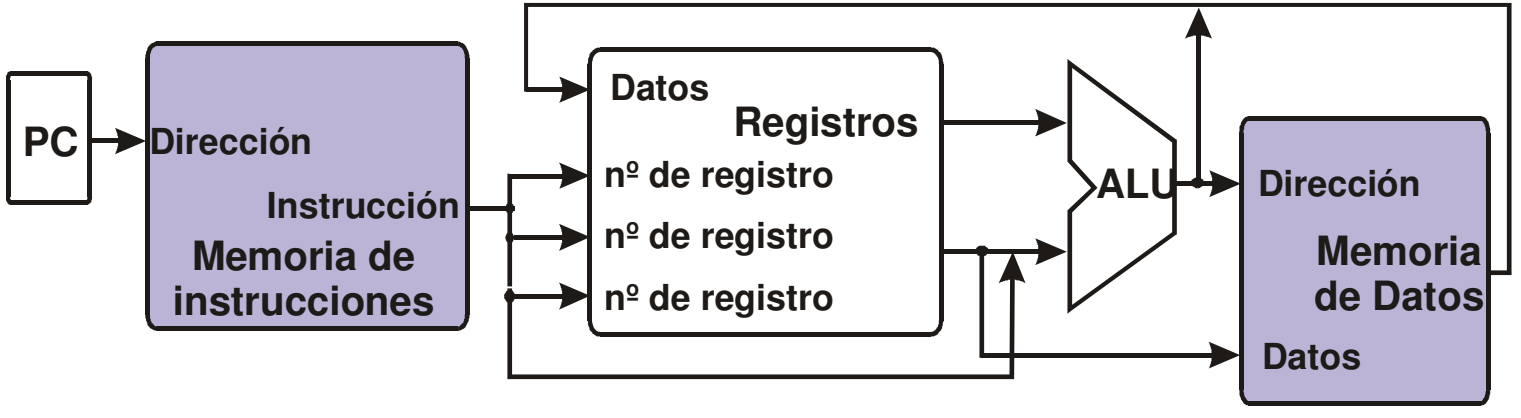


Diseño de un camino de datos sencillo

- Procesador monociclo:
  - √ Comenzaremos por los elementos básicos y sus funciones asociadas
  - √ Veremos los elementos necesarios para implementarlos
  - √ Veremos un conjunto de instrucciones básico
  - √ Veremos como implementar estas instrucciones

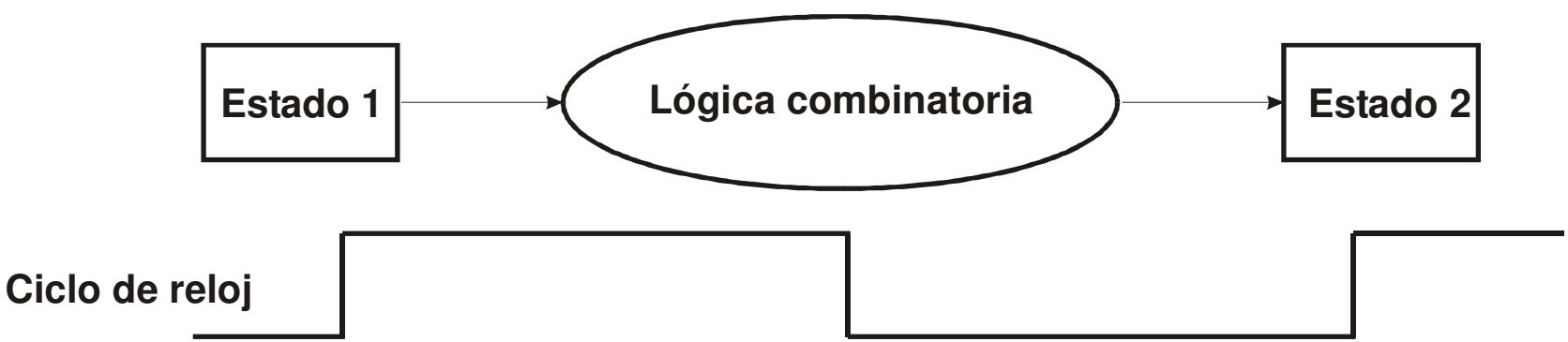
Estructura básica de un procesador

- 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



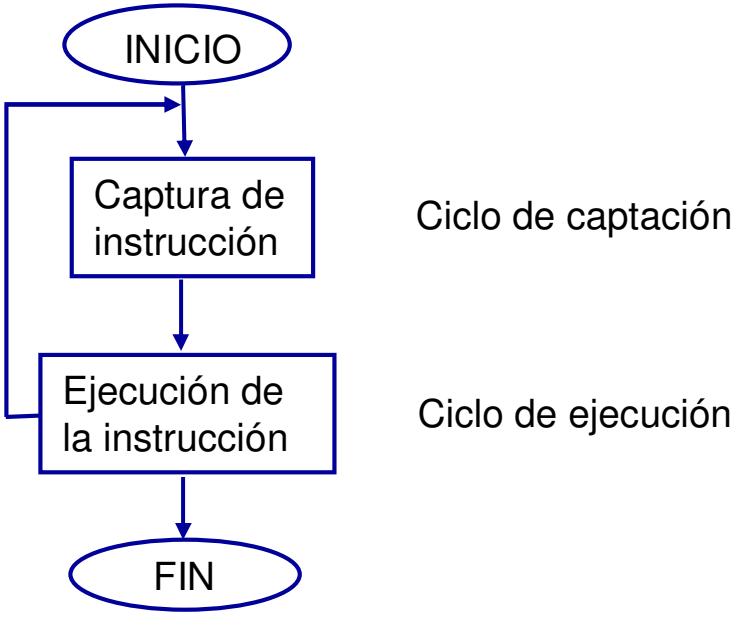
Estructura básica de un procesador

- Metodología de sincronización
  - √ Las unidades funcionales se clasifican en dos tipos: combinacionales y secuenciales
  - √ La metodología de sincronización define cuándo pueden leerse y escribirse la diferentes señales
  - √ Asumimos sincronización por flancos



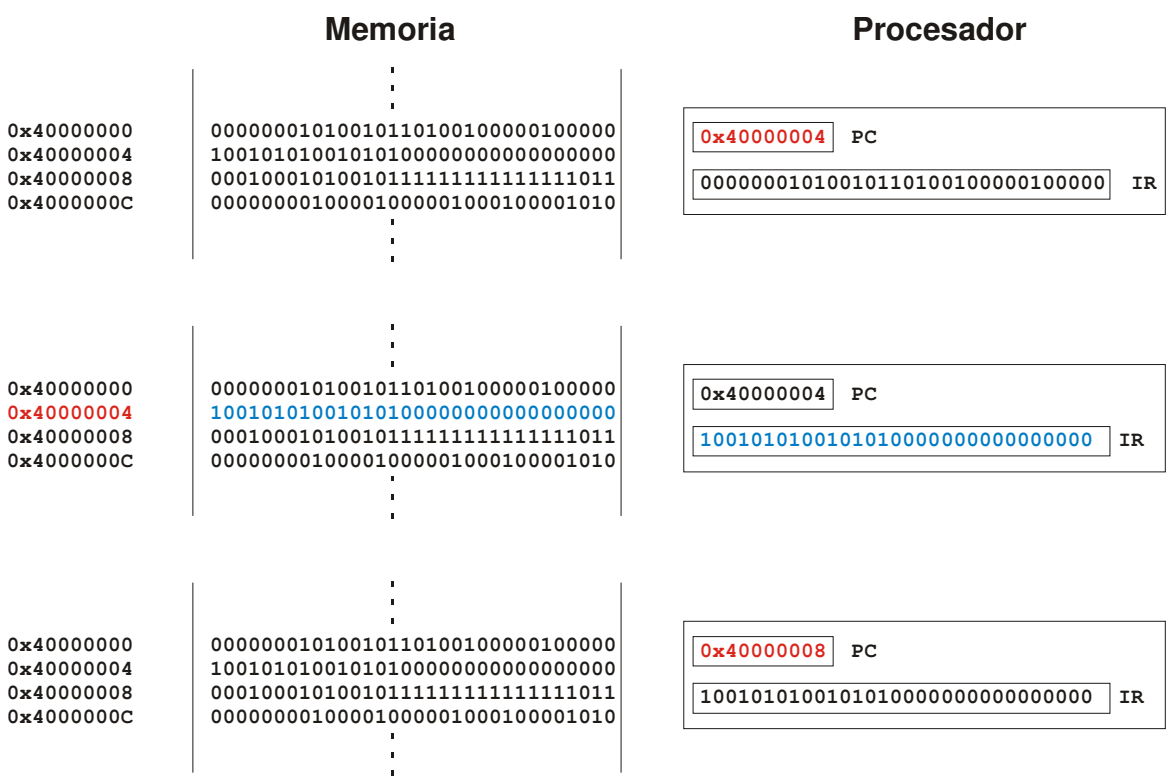
El ciclo de instrucción

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



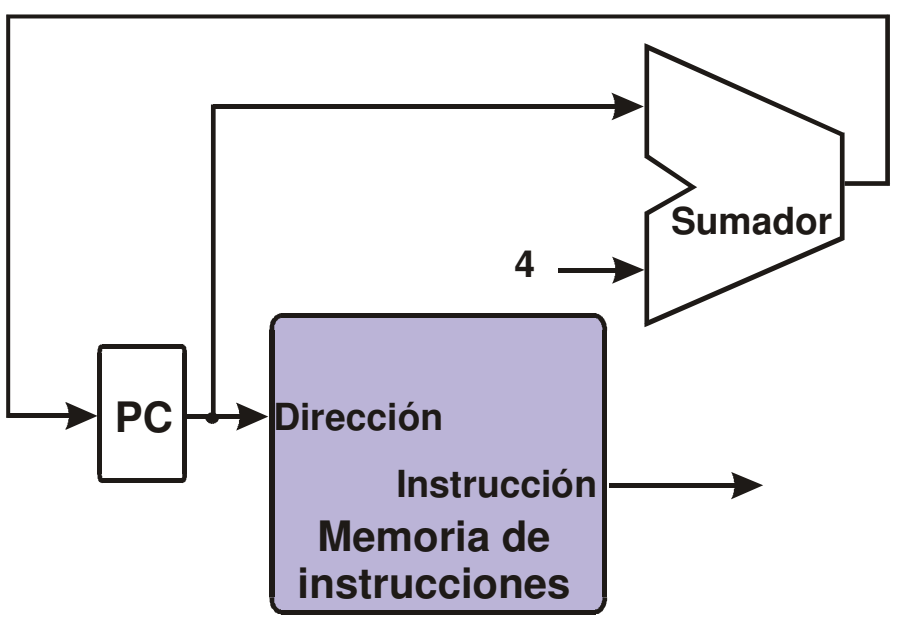
El ciclo de instrucción

- Para el subconjunto de instrucciones MIPS, los dos primeros pasos son idénticos:
  - √ Usar el contenido del PC para cargar, desde la memoria que contiene el código, la siguiente instrucción
  - √ Leer uno o dos registros, utilizando para ello los campos de la instrucción específicos para seleccionarlos



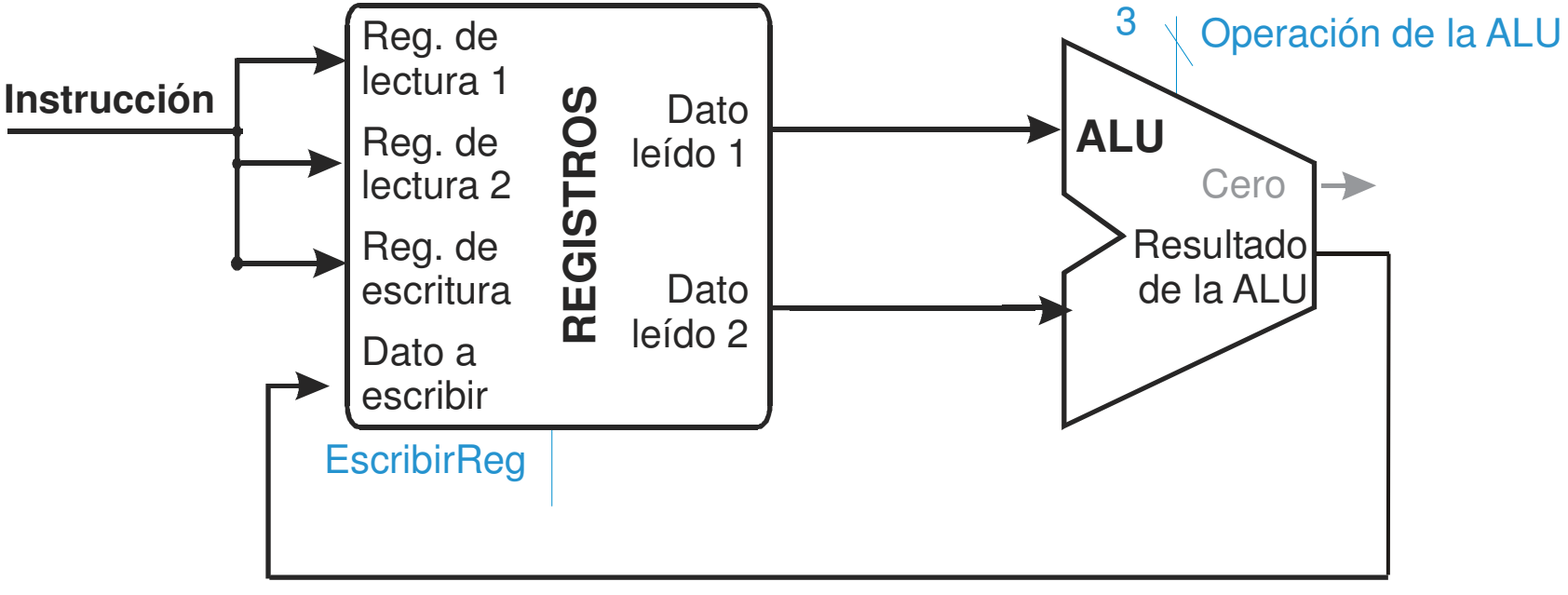
Diseño de un camino de datos sencillo

- Contador de programa:
  - √ Cada instrucción está en una dirección de memoria dada
  - √ Almacenamos la dirección en el registro PC
  - √ Tras procesar una instrucción avanzamos el contador hasta la siguiente instrucción



Diseño de un camino de datos sencillo

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

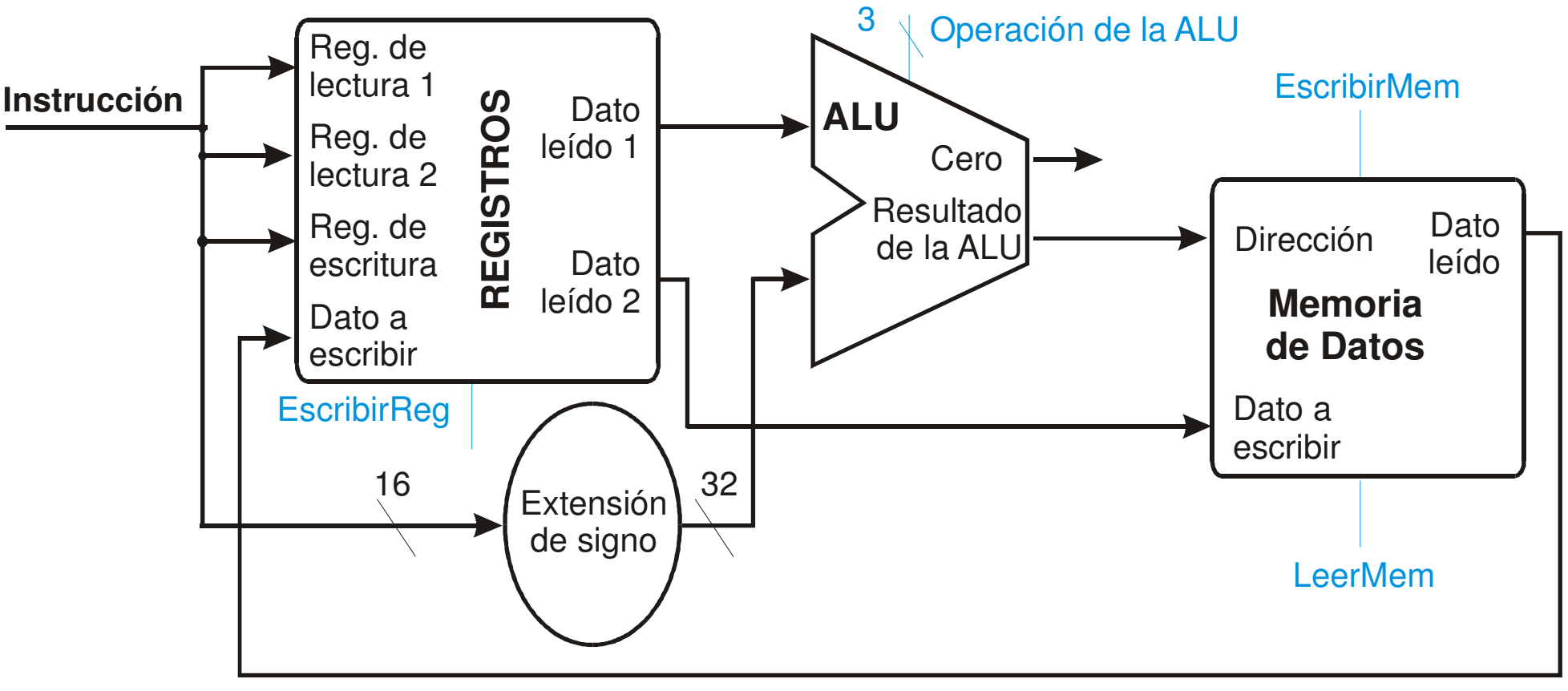


Diseño de un camino de datos sencillo

- Acceso a memoria:
  - √ Instrucciones lw y sw
  - √ La dirección se indica con un registro más un desplazamiento de 16 bits con signo
  - √ El desplazamiento se extiende a 32 bits

lw $t0, 8($s0)	100011	10000	01000	0000000000001000
sw $t0, 32($s0)	101011	10000	01000	0000000000100000

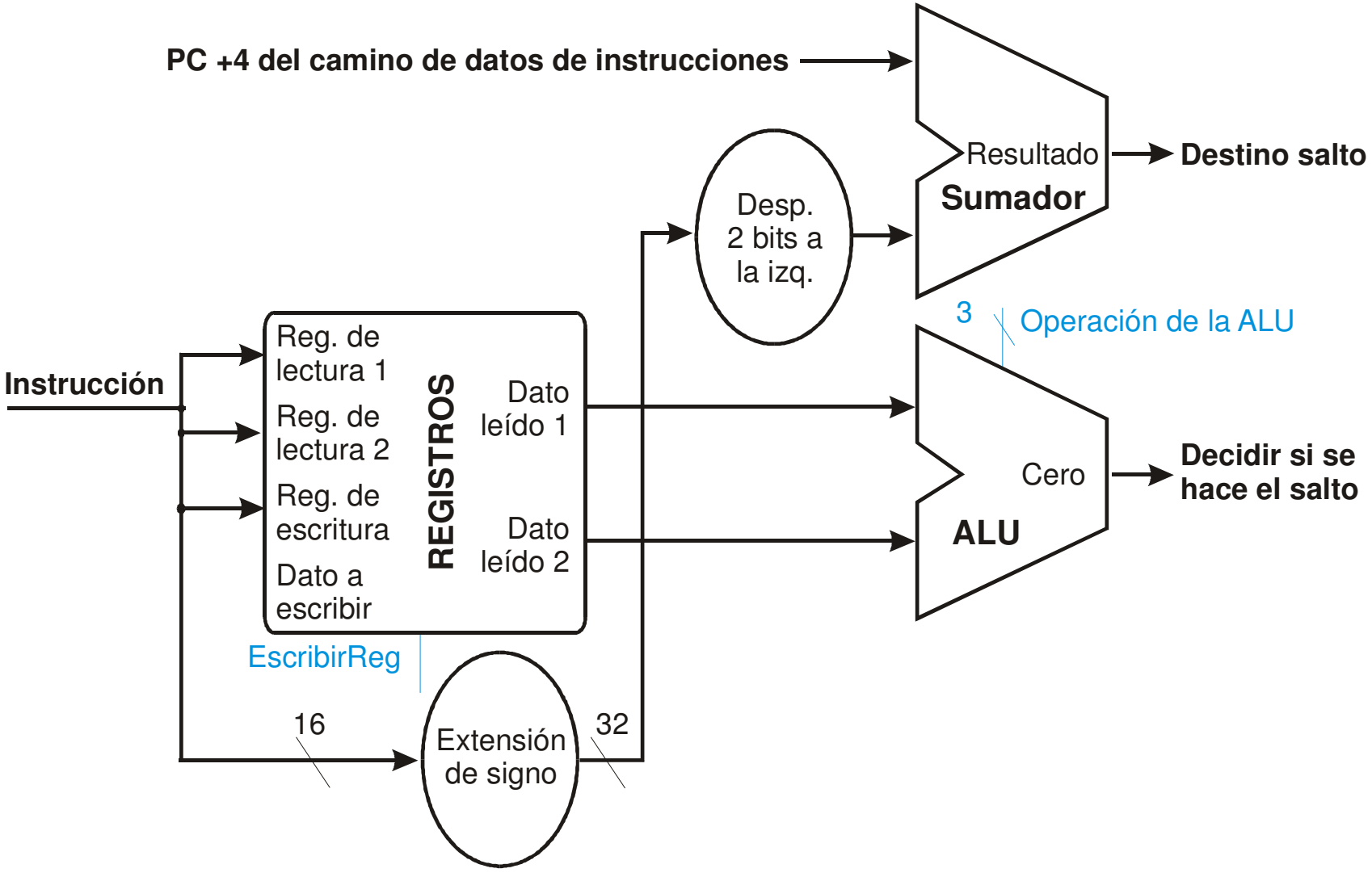
Diseño de un camino de datos sencillo



Diseño de un camino de datos sencillo

- Saltos condicionales:
  - √ Evalúan si dos registros contienen ó no el mismo valor
  - √ Si la condición se cumple aplican el salto
  - √ El salto es relativo con signo
  - √ Los 16 bits se extienden a 32 y se desplazan 2 posiciones a la izquierda para direccionar sólo palabras completas
  - √ El PC ya se ha actualizado a PC + 4

Diseño de un camino de datos sencillo

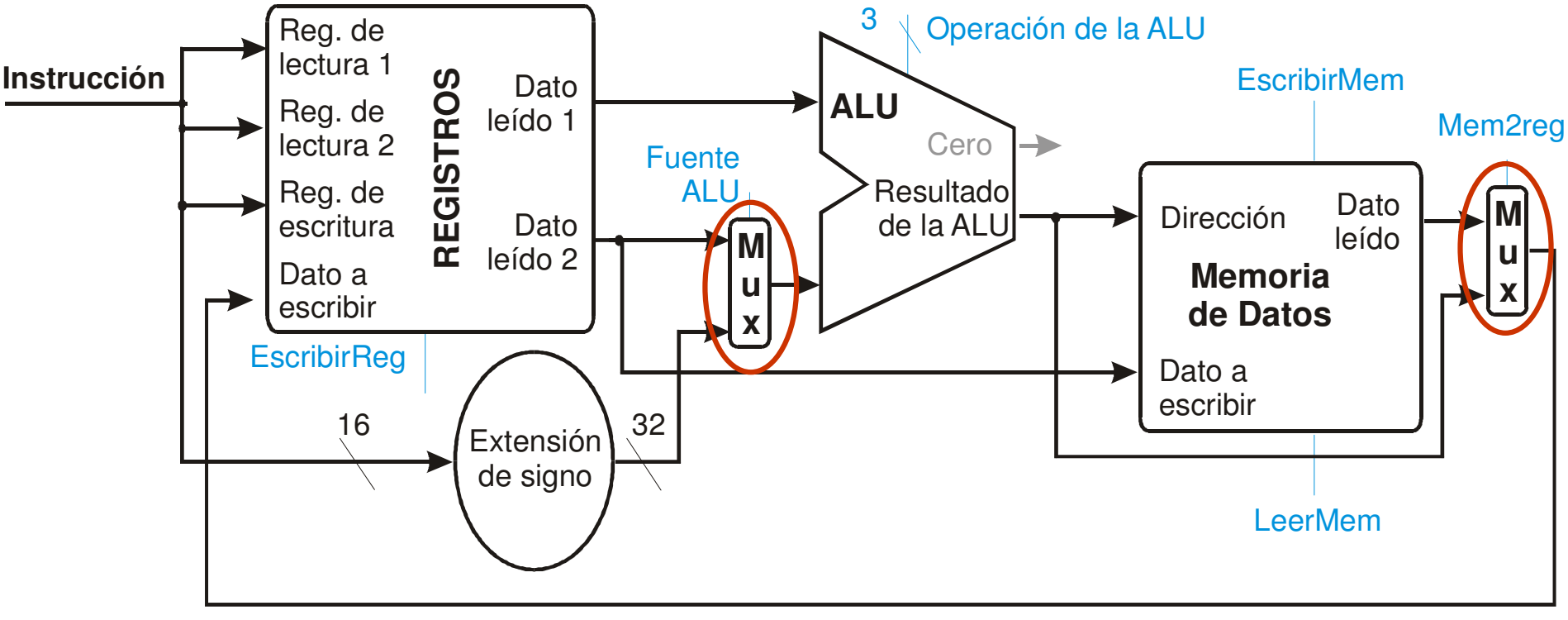


Diseño de un camino de datos sencillo

- Todo junto:
  - Para construir el camino de datos hemos de combinar los elementos explicados anteriormente
  - Intentaremos reutilizar parte del hardware
  - El hardware compartido selecciona los datos mediante multiplexores
  - Parte del hardware no se podrá reutilizar y habrá que replicarlo

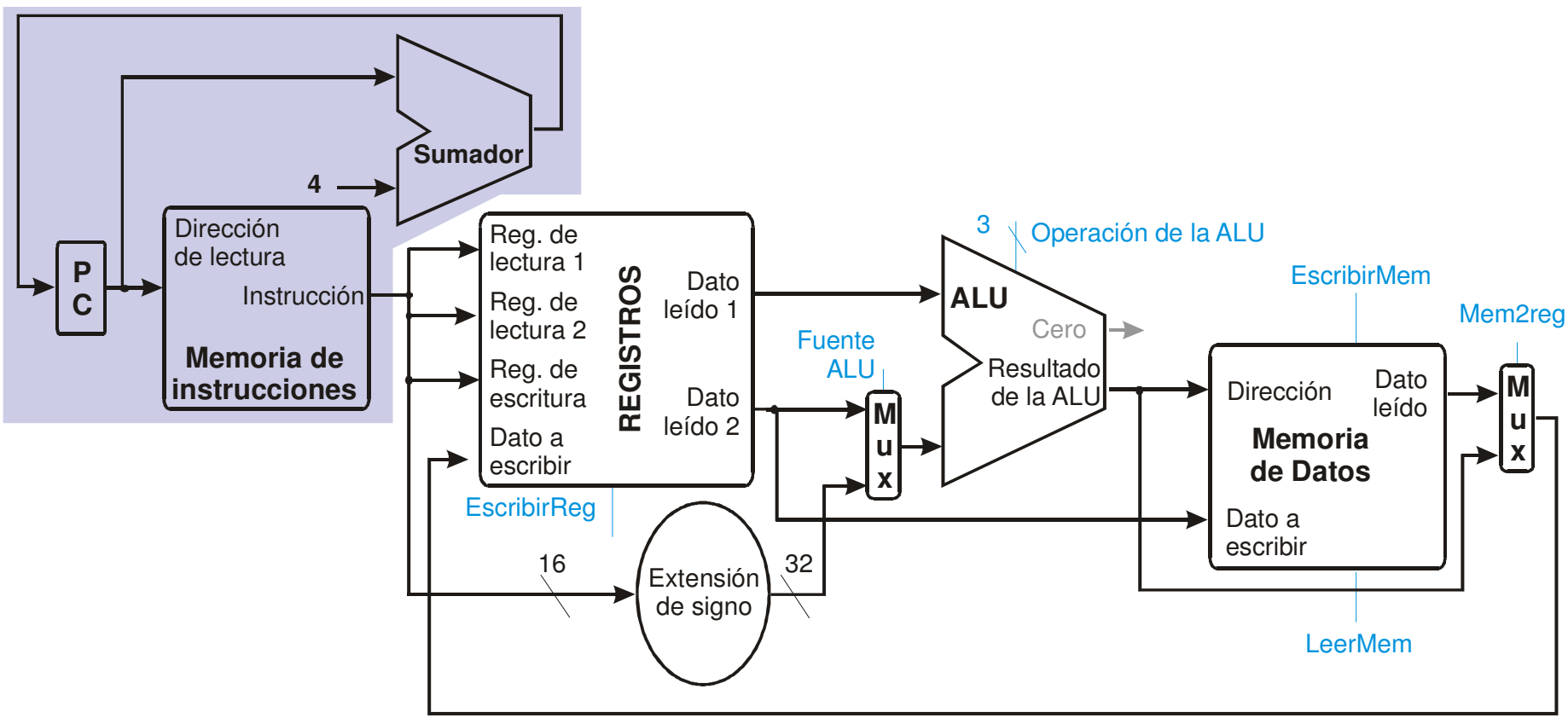
Diseño de un camino de datos sencillo

Aritmética + Acceso a Memoria



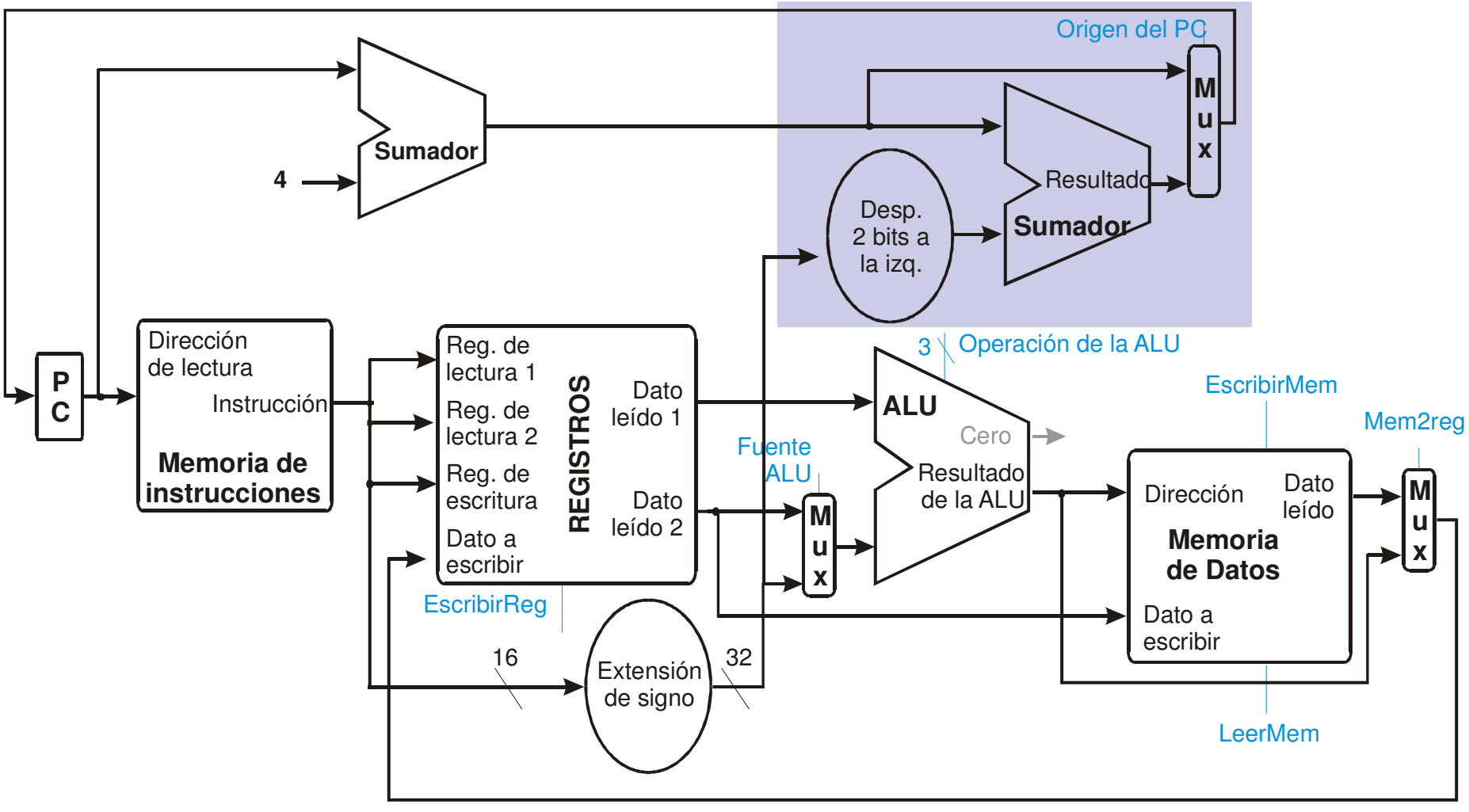
Diseño de un camino de datos sencillo

Incorporamos gestión de PC



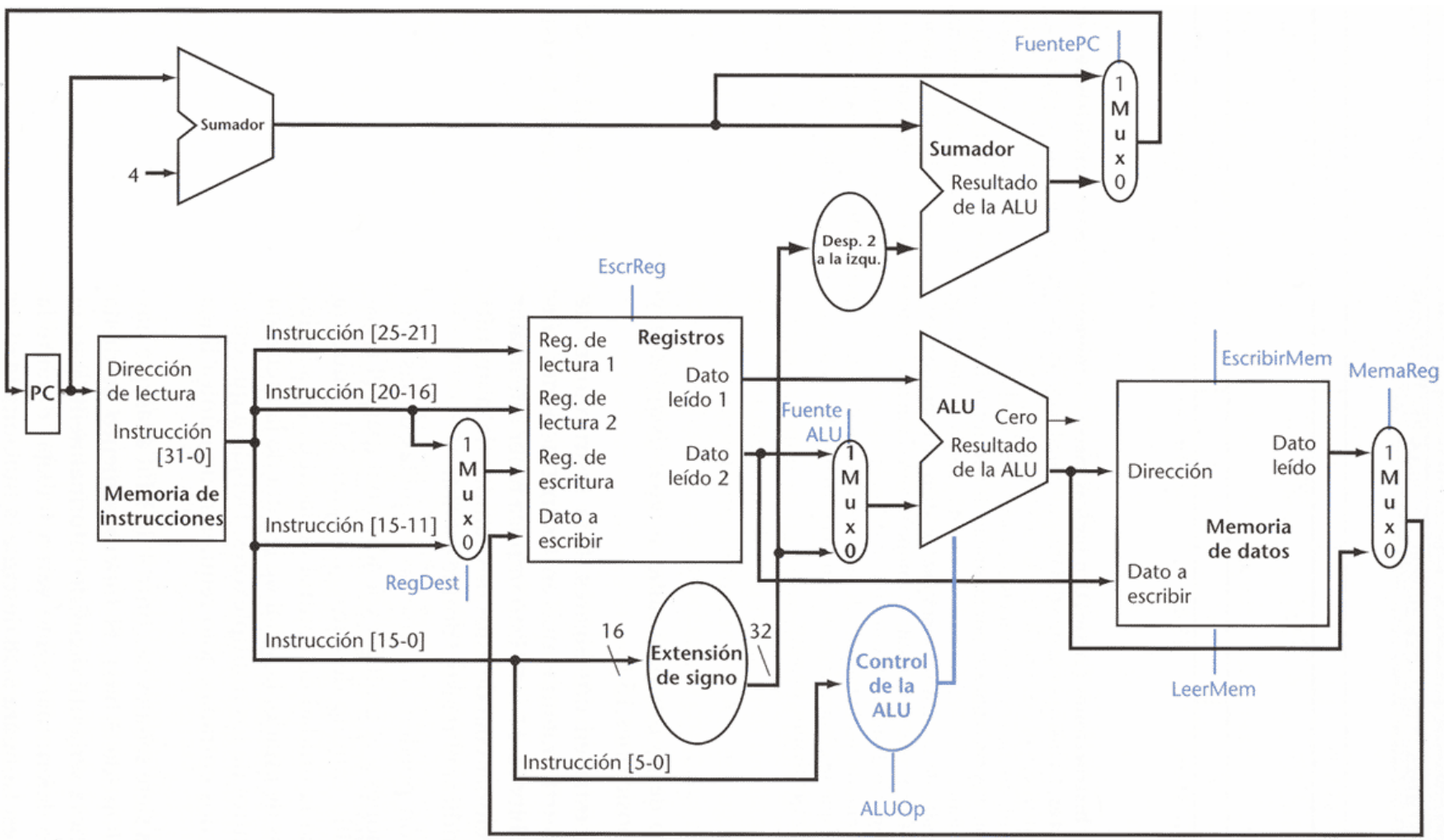
Diseño de un camino de datos sencillo

Añadimos saltos condicionales



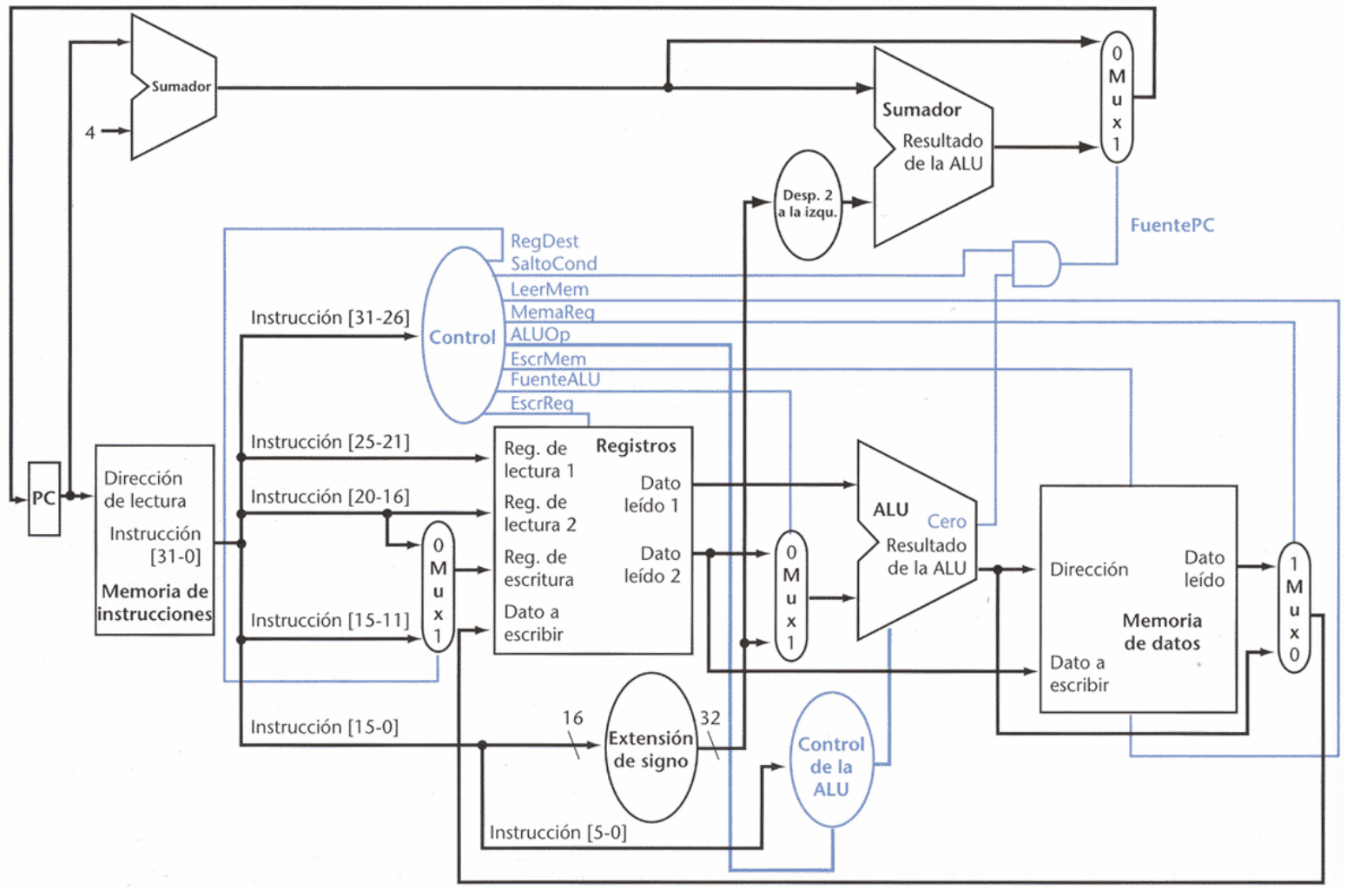
Diseño de la unidad de control

- Realización monociclo: señales de control



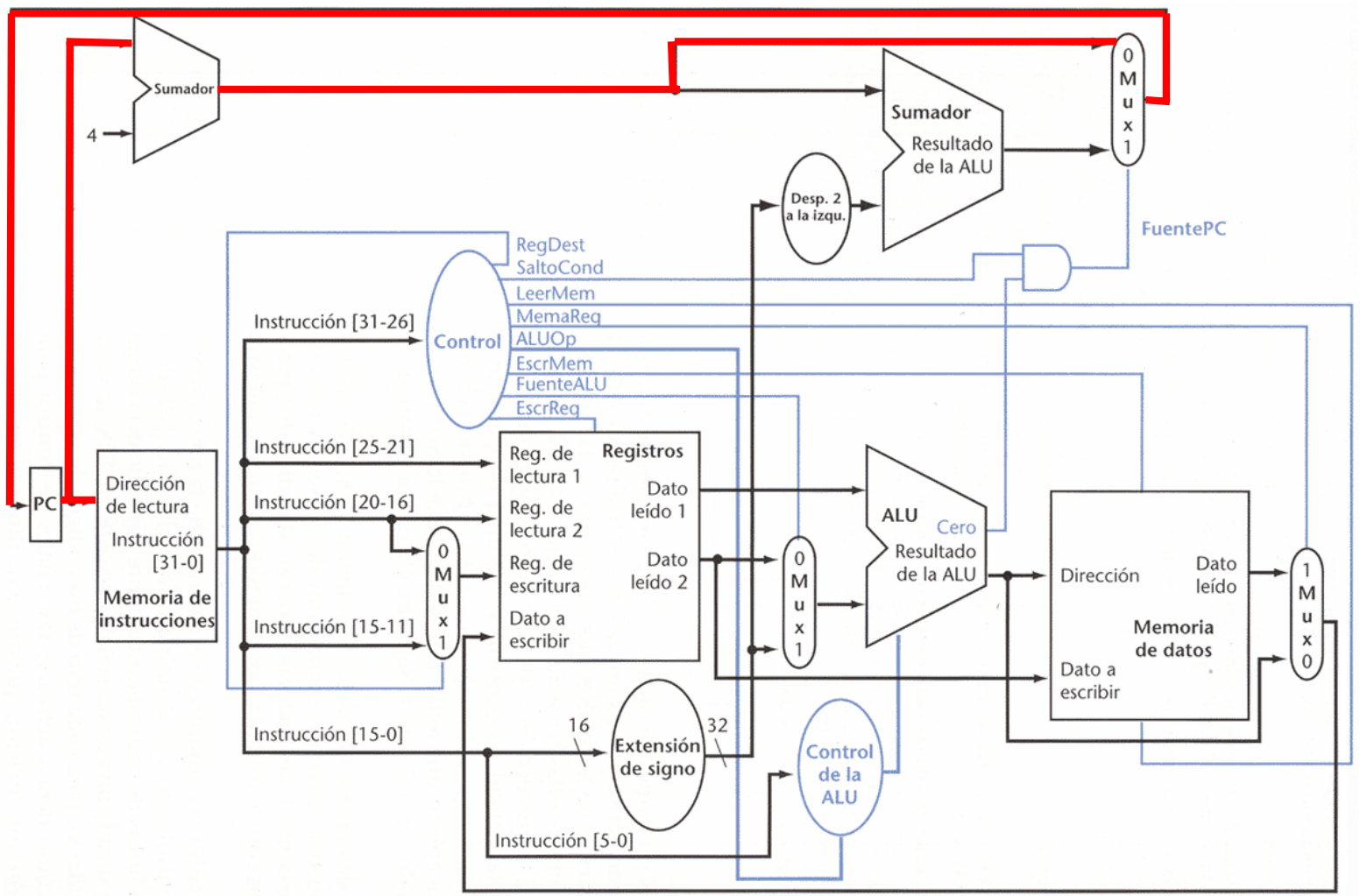
Diseño de la unidad de control

- Activación de las líneas determinada por el código de operación:



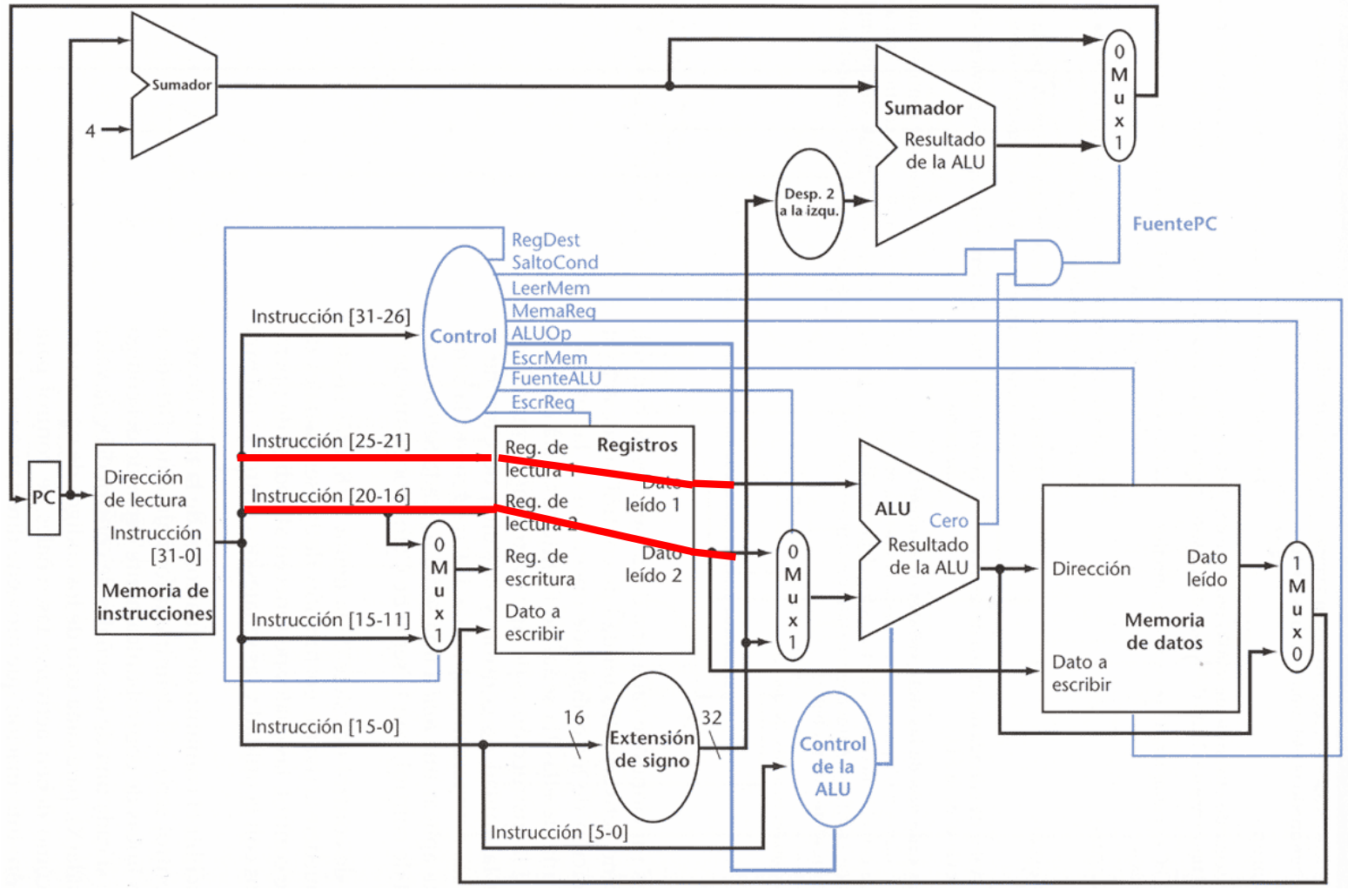
Diseño de la unidad de control

- Pasos en la realización de una instrucción tipo R (I):



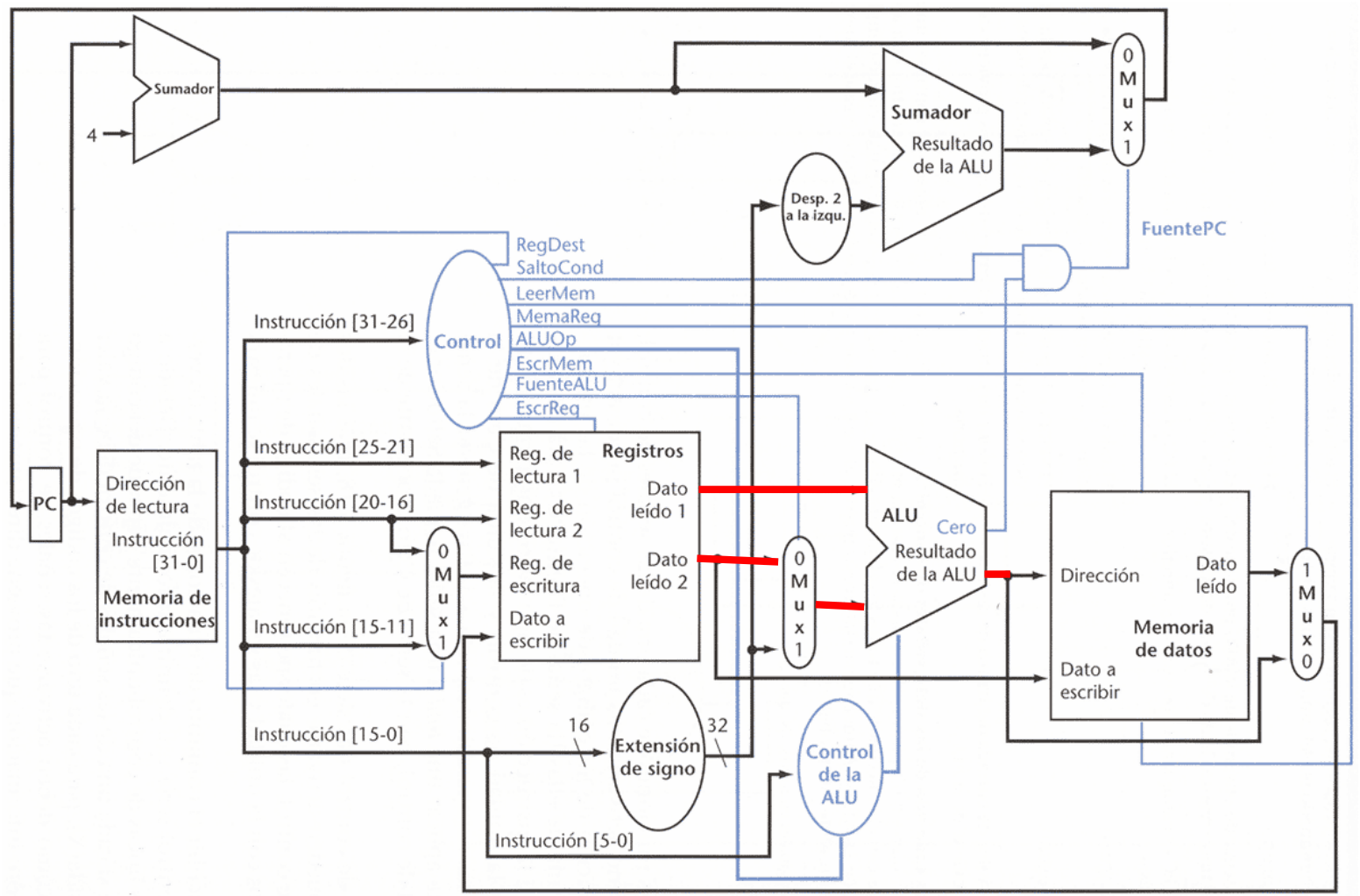
Diseño de la unidad de control

- Pasos en la realización de una instrucción tipo R (II):



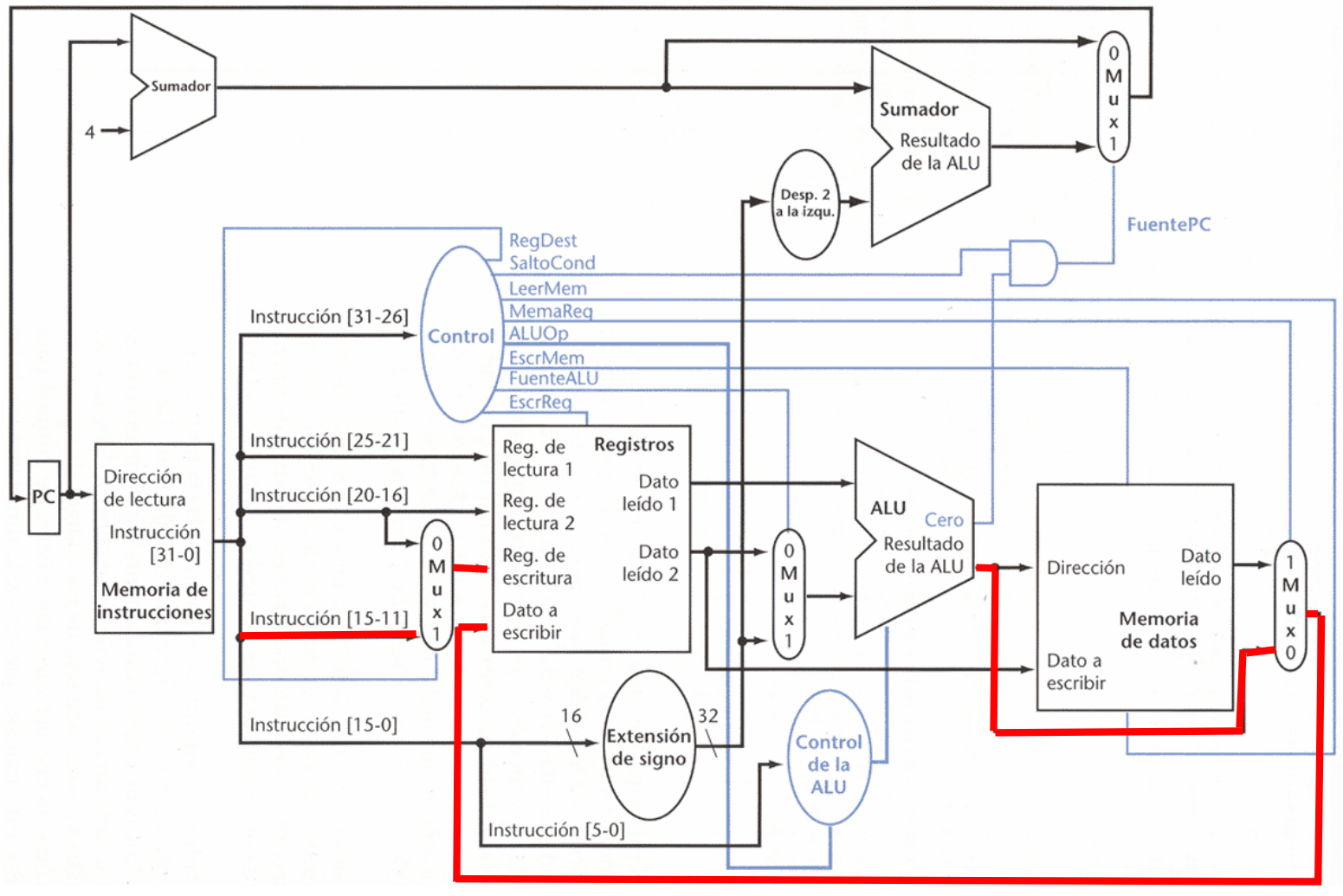
Diseño de la unidad de control

- Pasos en la realización de una instrucción tipo R (III):



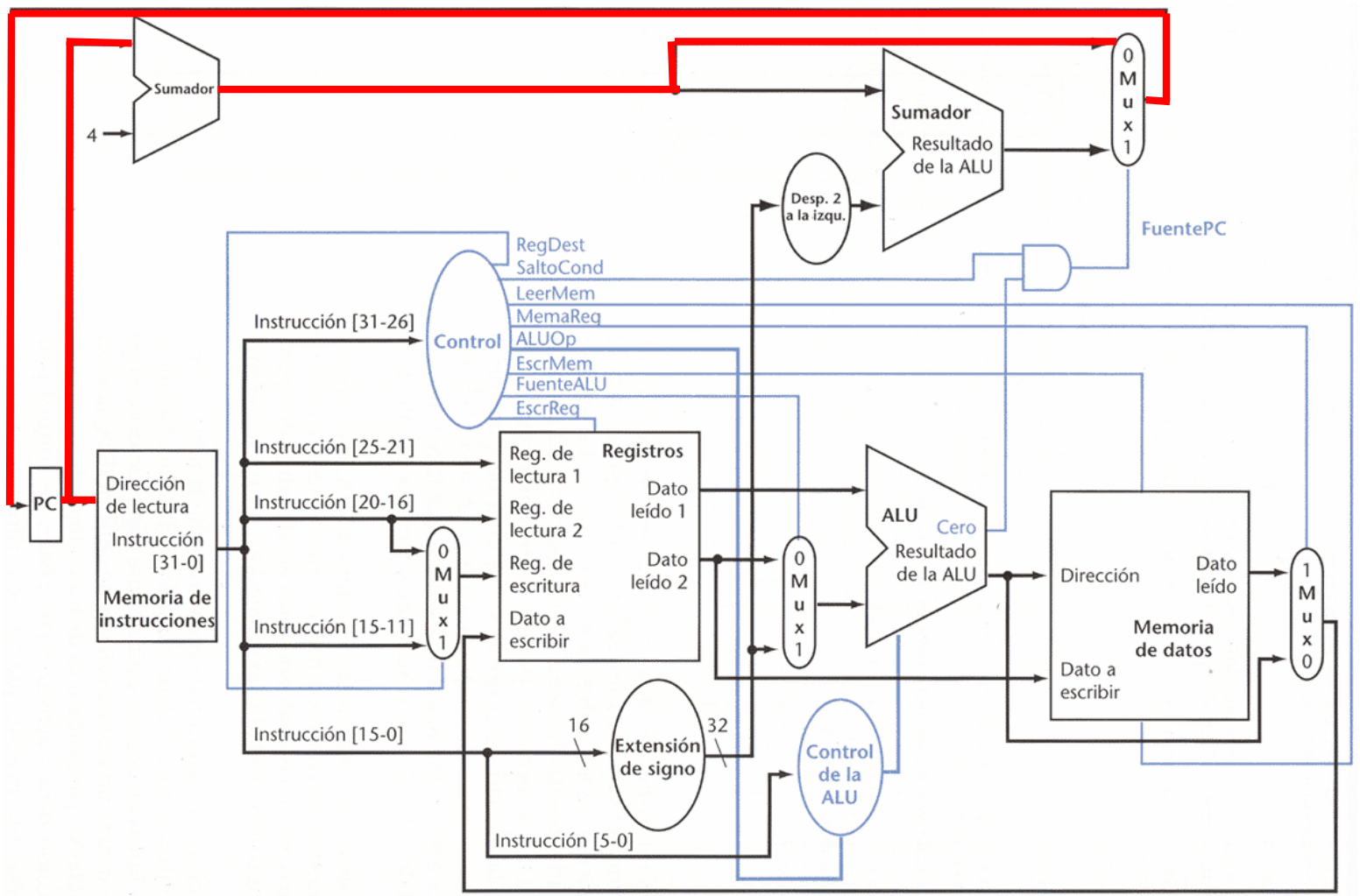
Diseño de la unidad de control

- Pasos en la realización de una instrucción tipo R (IV):



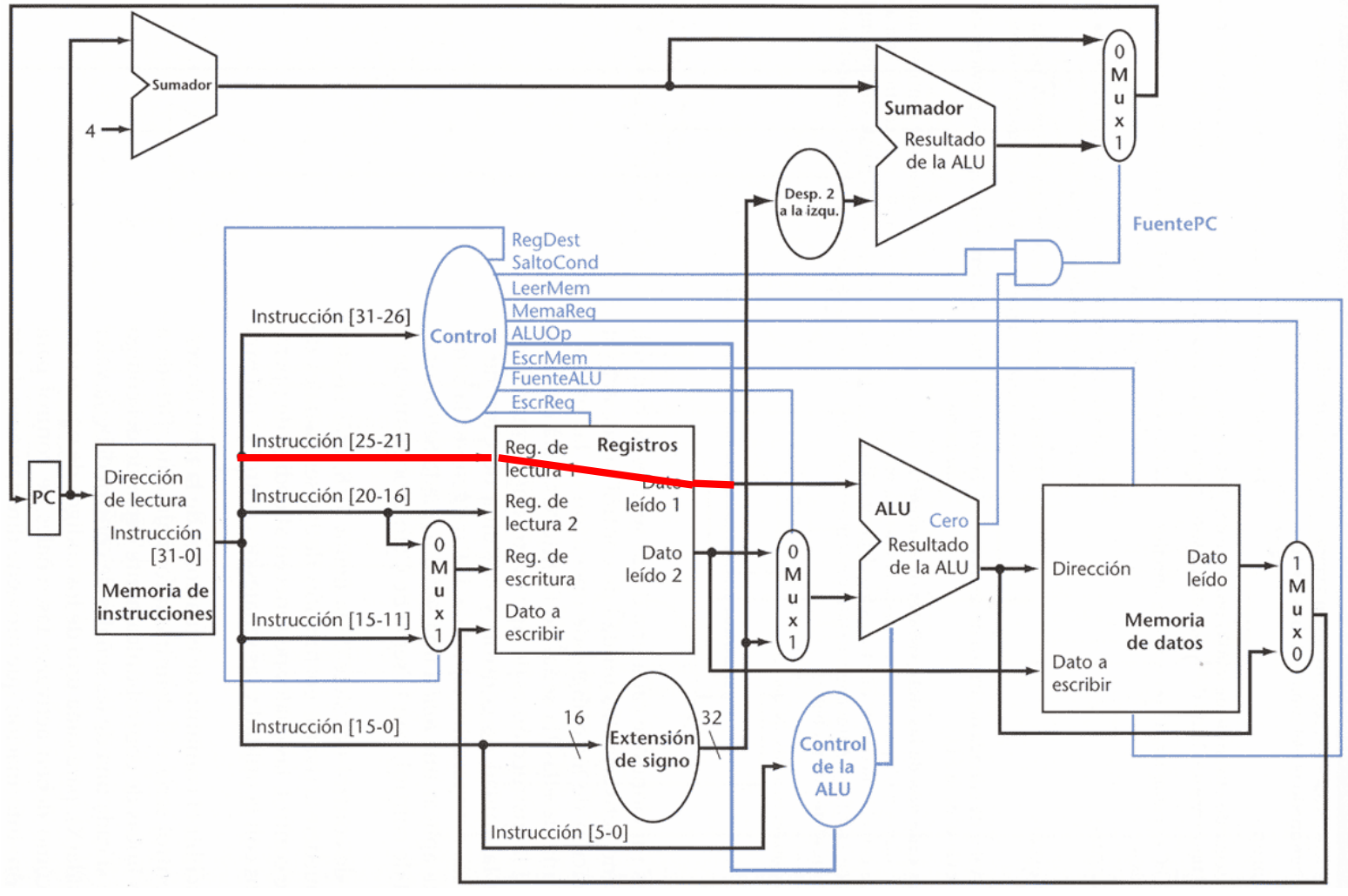
Diseño de la unidad de control

- Pasos en la realización de una instrucción lw (I):



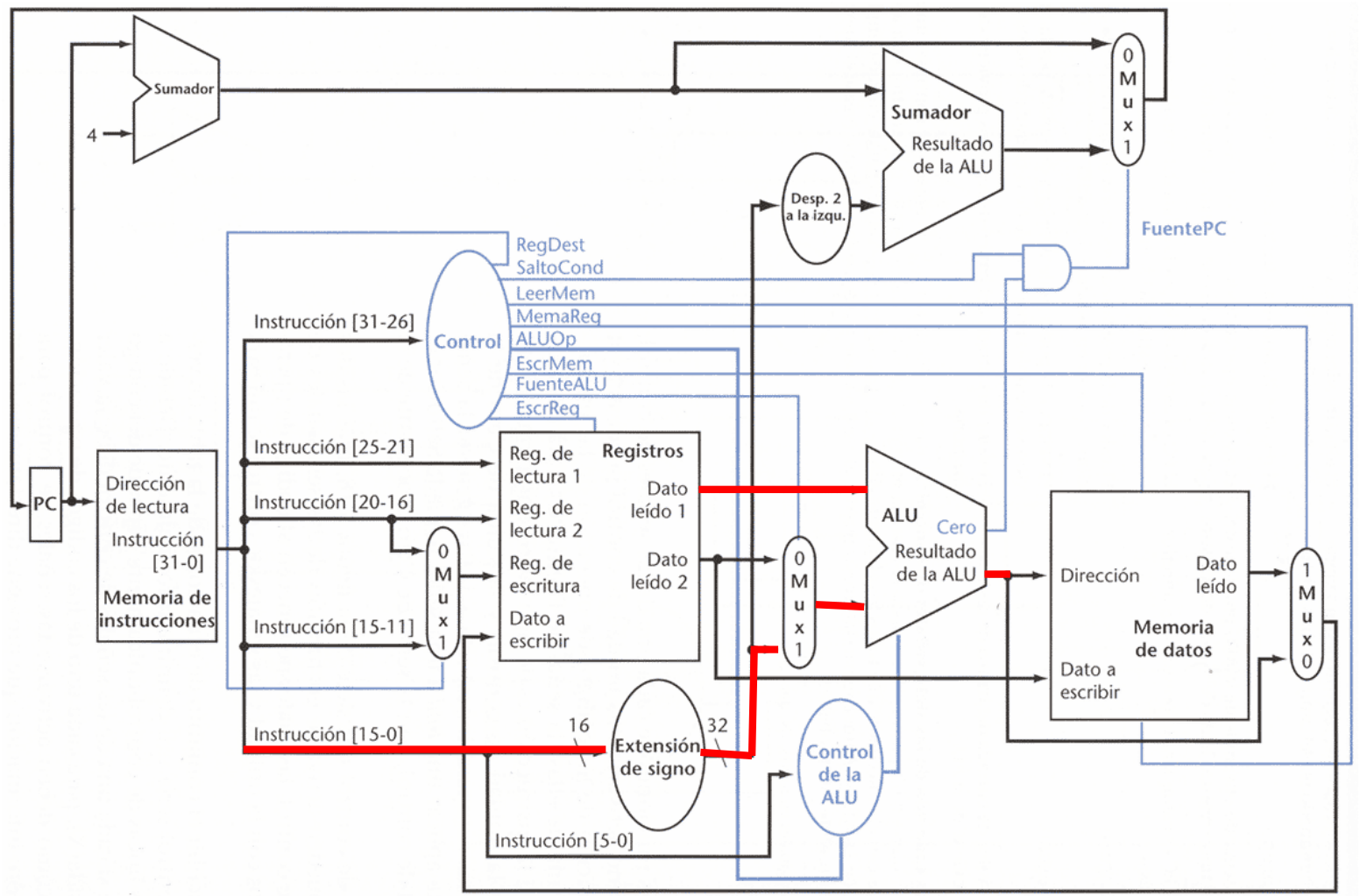
Diseño de la unidad de control

- Pasos en la realización de una instrucción lw (II):



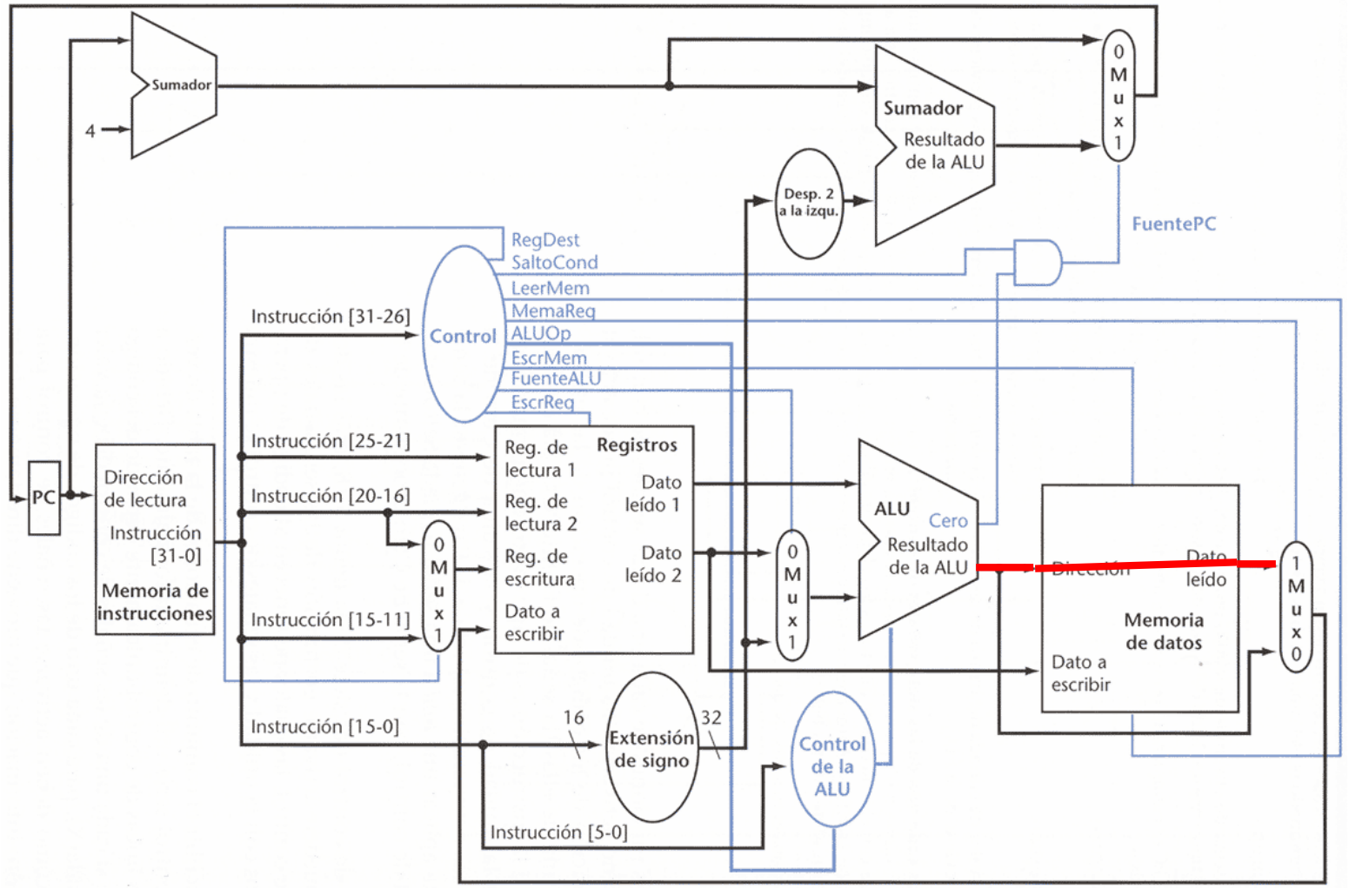
Diseño de la unidad de control

- Pasos en la realización de una instrucción lw (III):



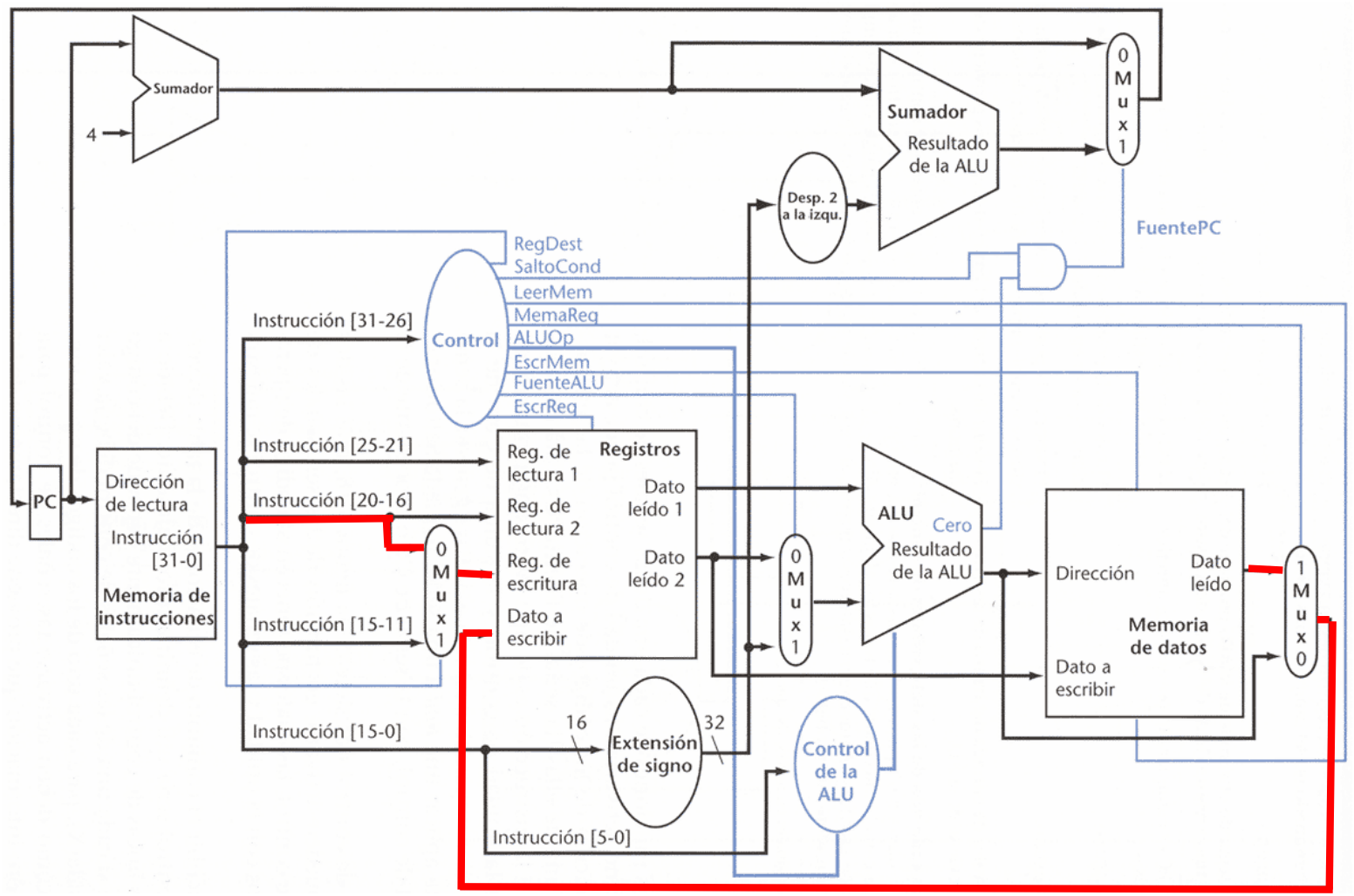
Diseño de la unidad de control

- Pasos en la realización de una instrucción lw (IV):



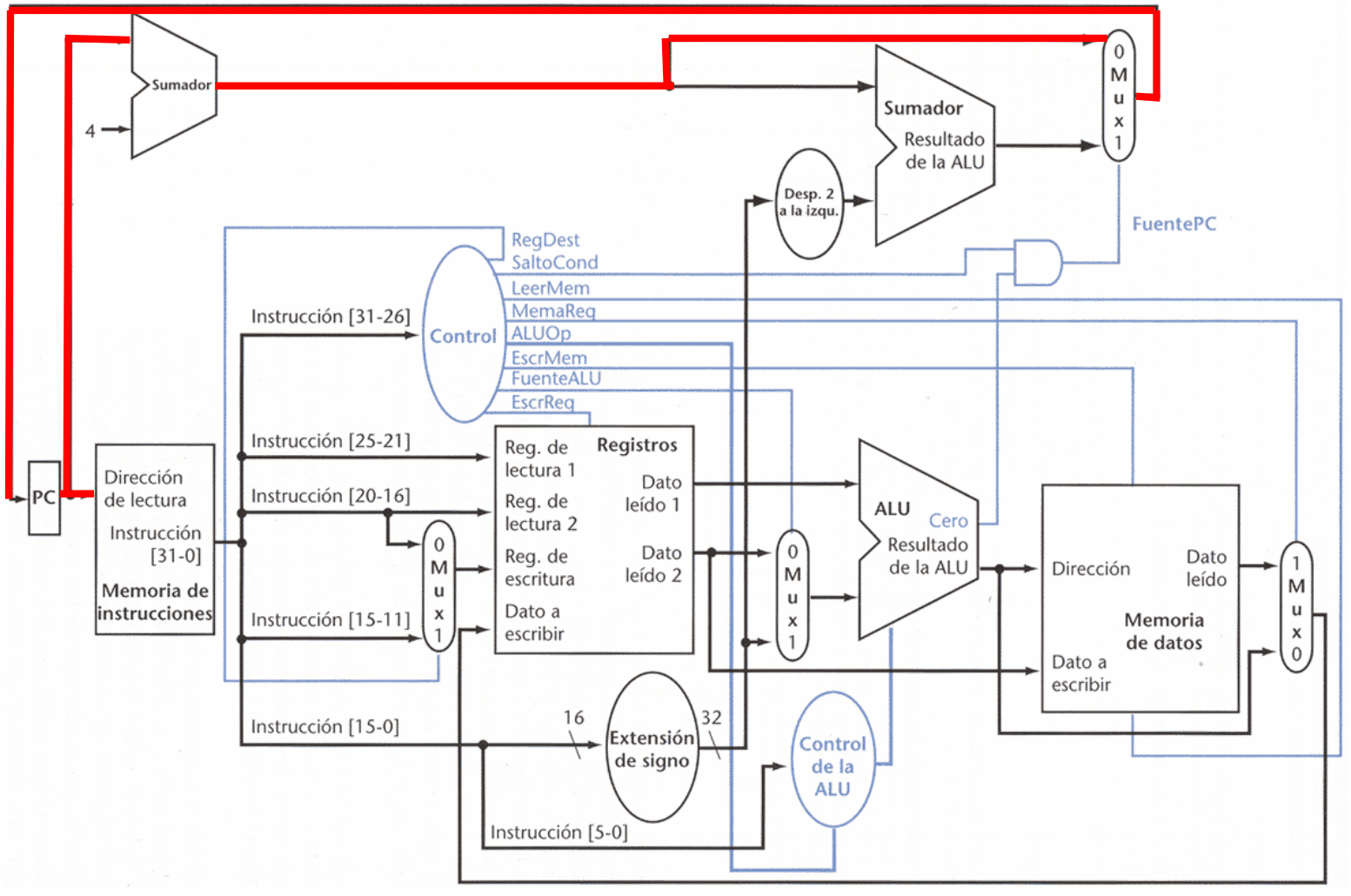
Diseño de la unidad de control

- Pasos en la realización de una instrucción lw (V):



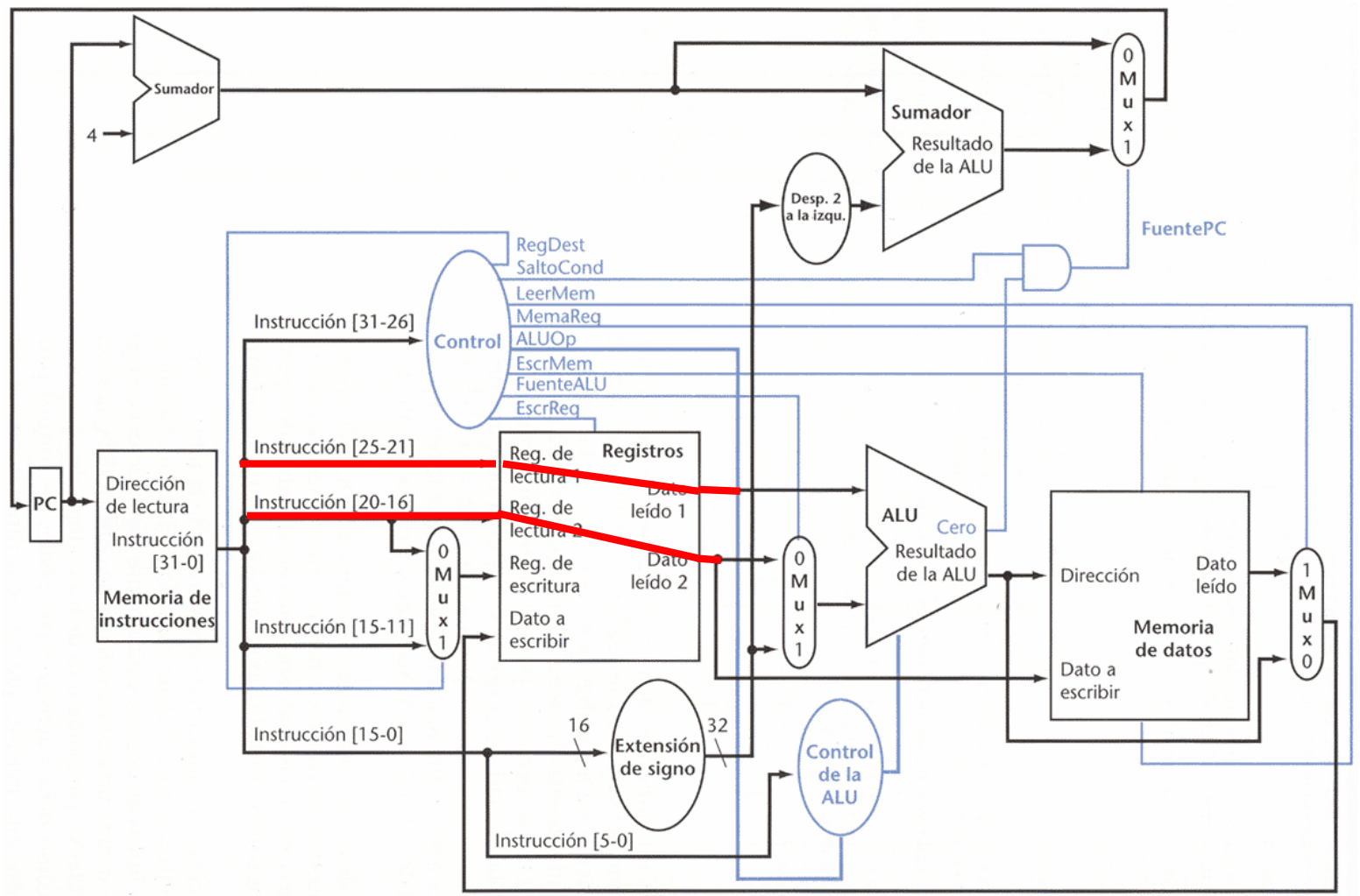
Diseño de la unidad de control

- Pasos en la realización de una instrucción sw (I):



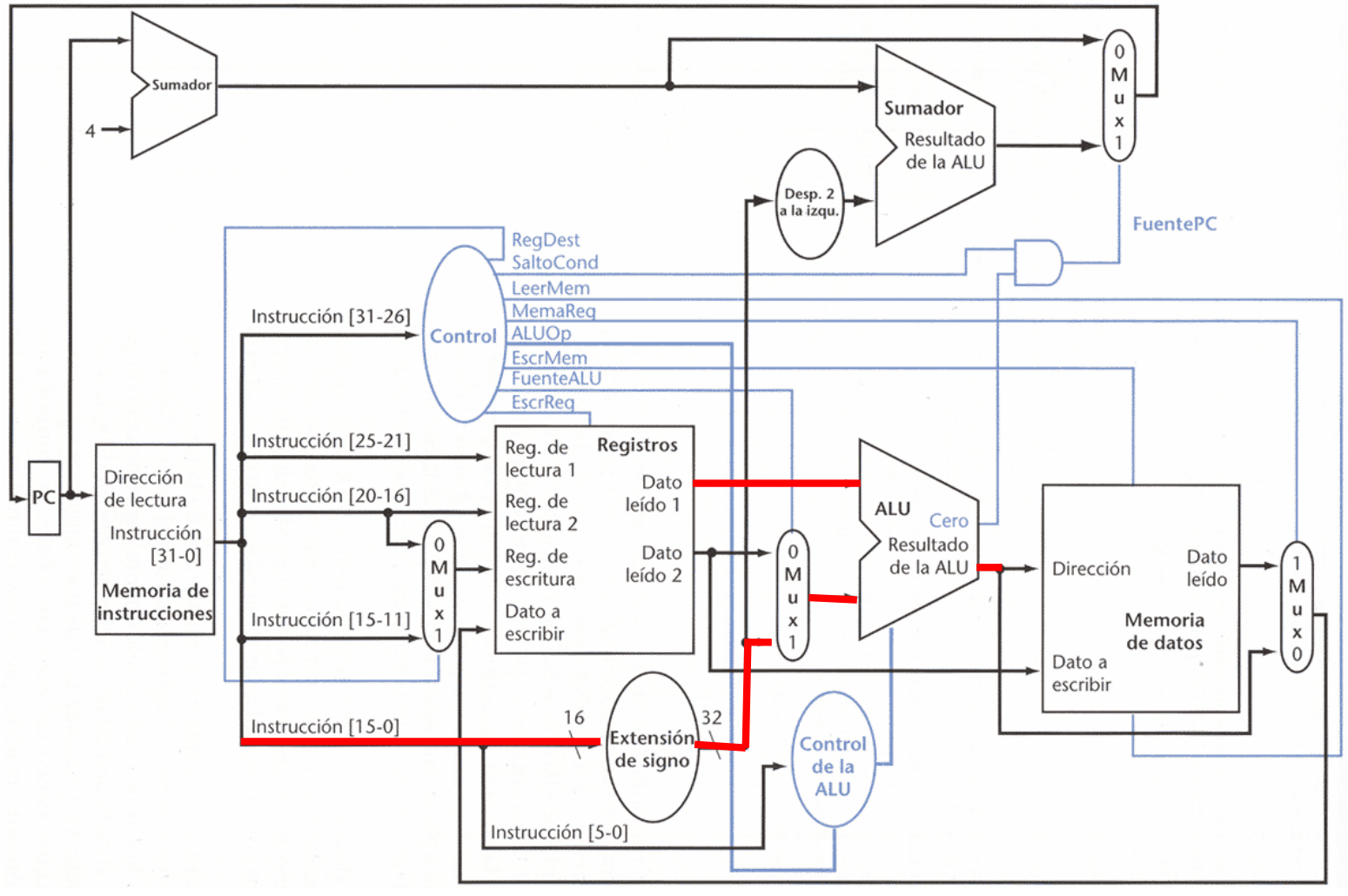
Diseño de la unidad de control

- Pasos en la realización de una instrucción sw (II):



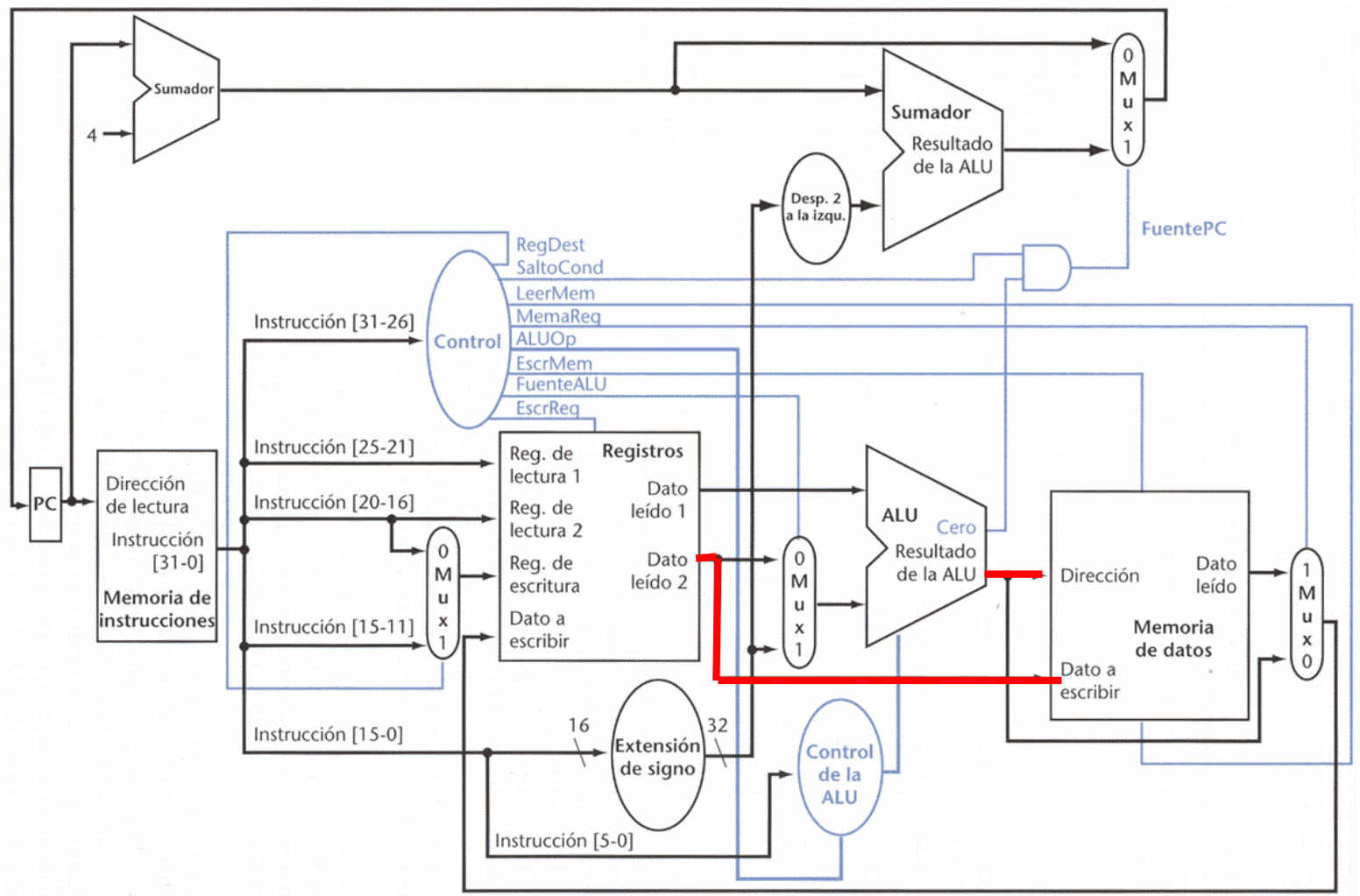
Diseño de la unidad de control

- Pasos en la realización de una instrucción sw (III):



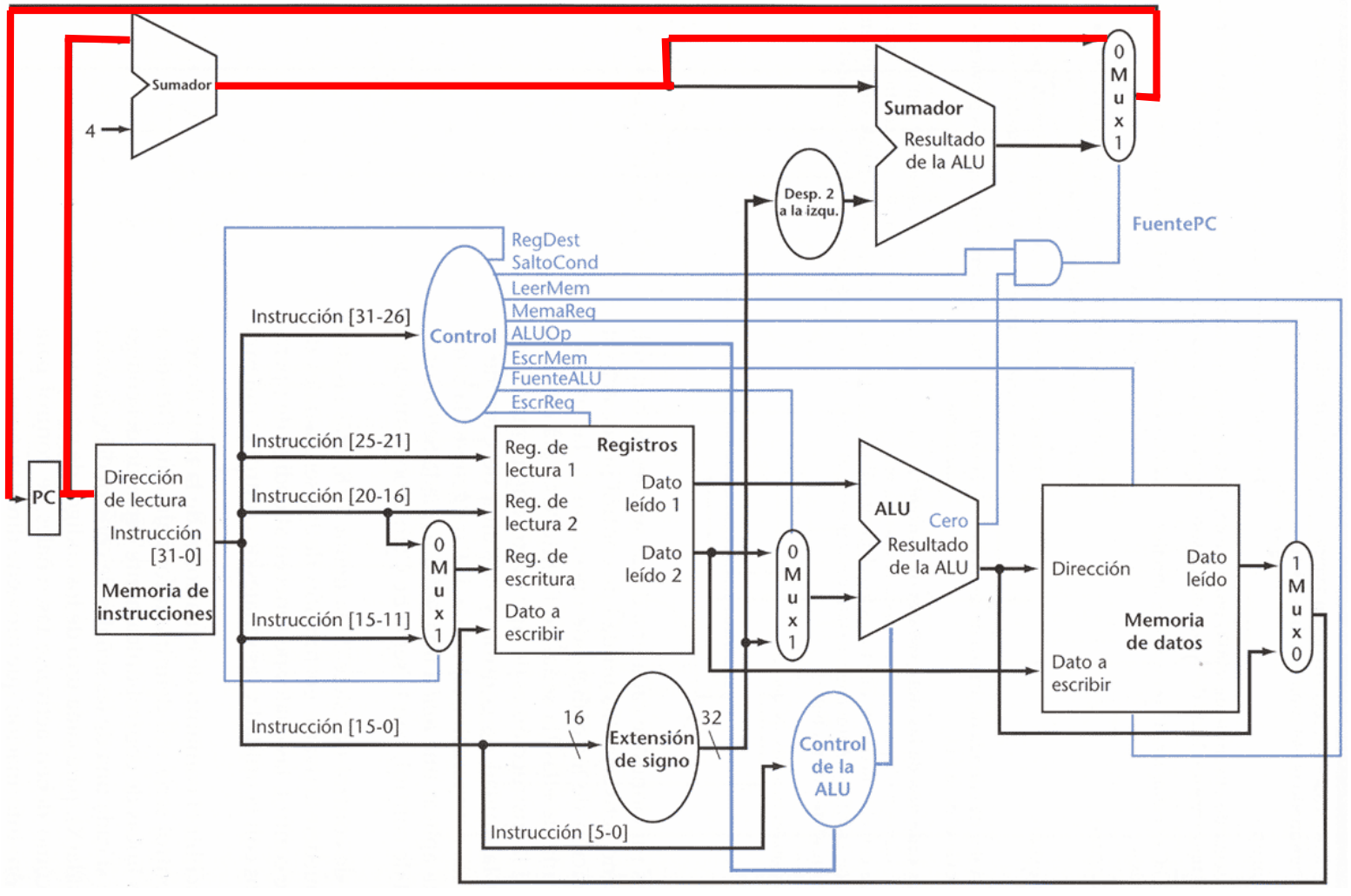
Diseño de la unidad de control

- Pasos en la realización de una instrucción sw (IV):



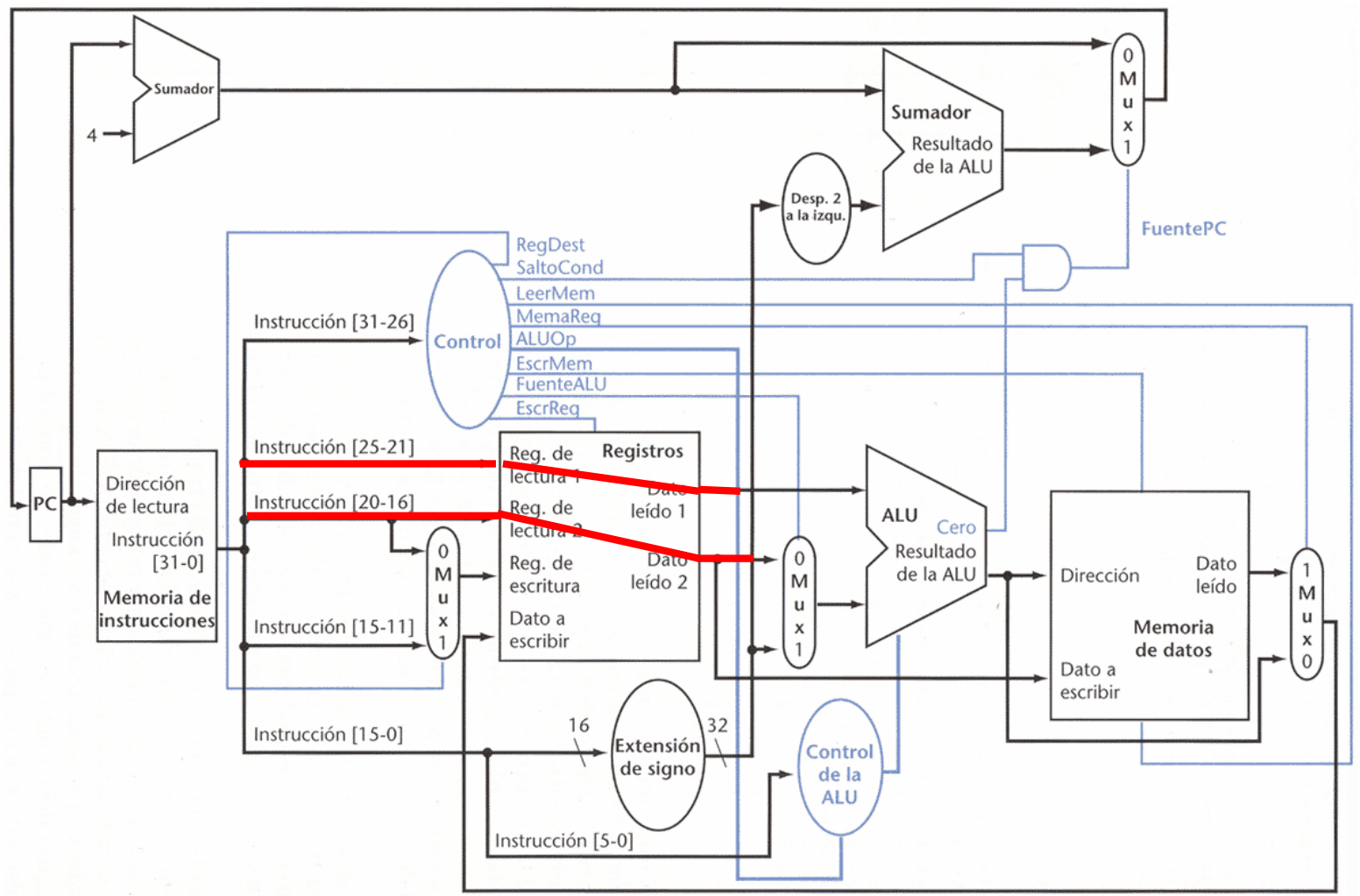
Diseño de la unidad de control

- Pasos en la realización de una instrucción beq (I):



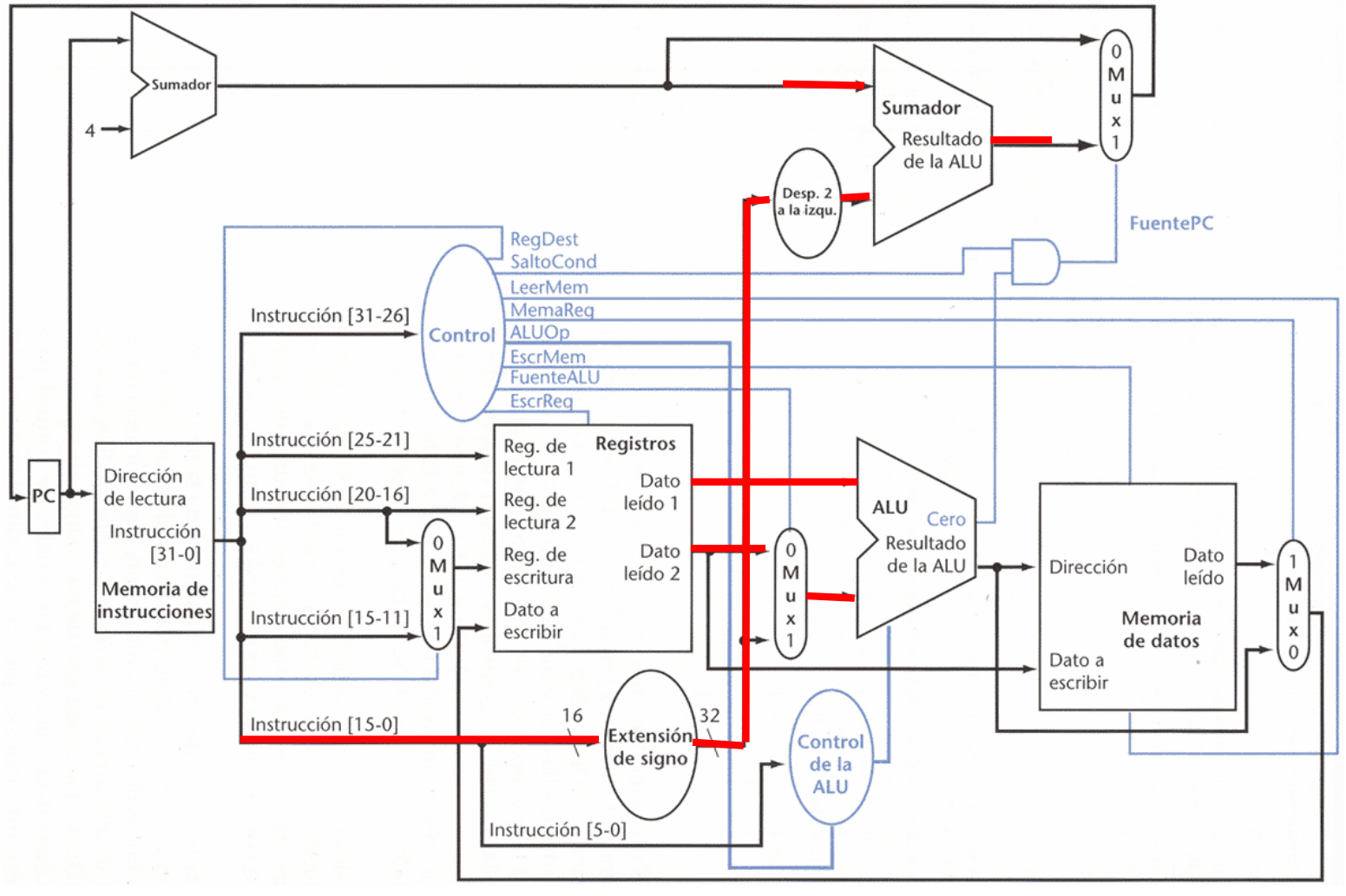
Diseño de la unidad de control

- Pasos en la realización de una instrucción beq (II):



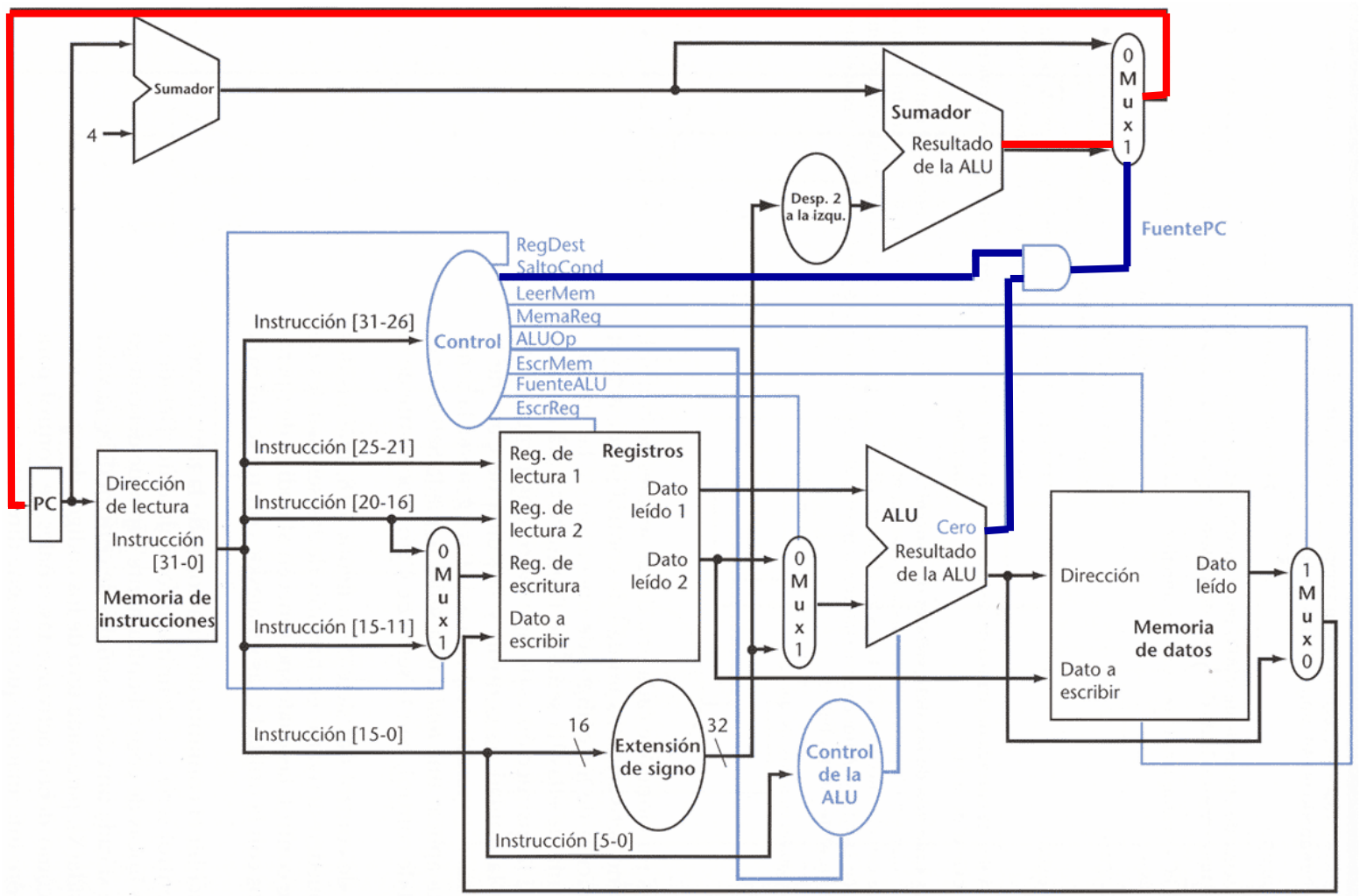
Diseño de la unidad de control

- Pasos en la realización de una instrucción beq (III):



Diseño de la unidad de control

- Pasos en la realización de una instrucción beq (IV):



Diseño de la unidad de control

- La función de control para una realización monociclo está especificada por la siguiente tabla de verdad:

	señal	Formato R	Lw	Sw	beq
entradas	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
salidas	RegDest	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

Ejercicio:

Mapas de Karnaugh

Implementación combinacional

Inconvenientes de la implementación monociclo

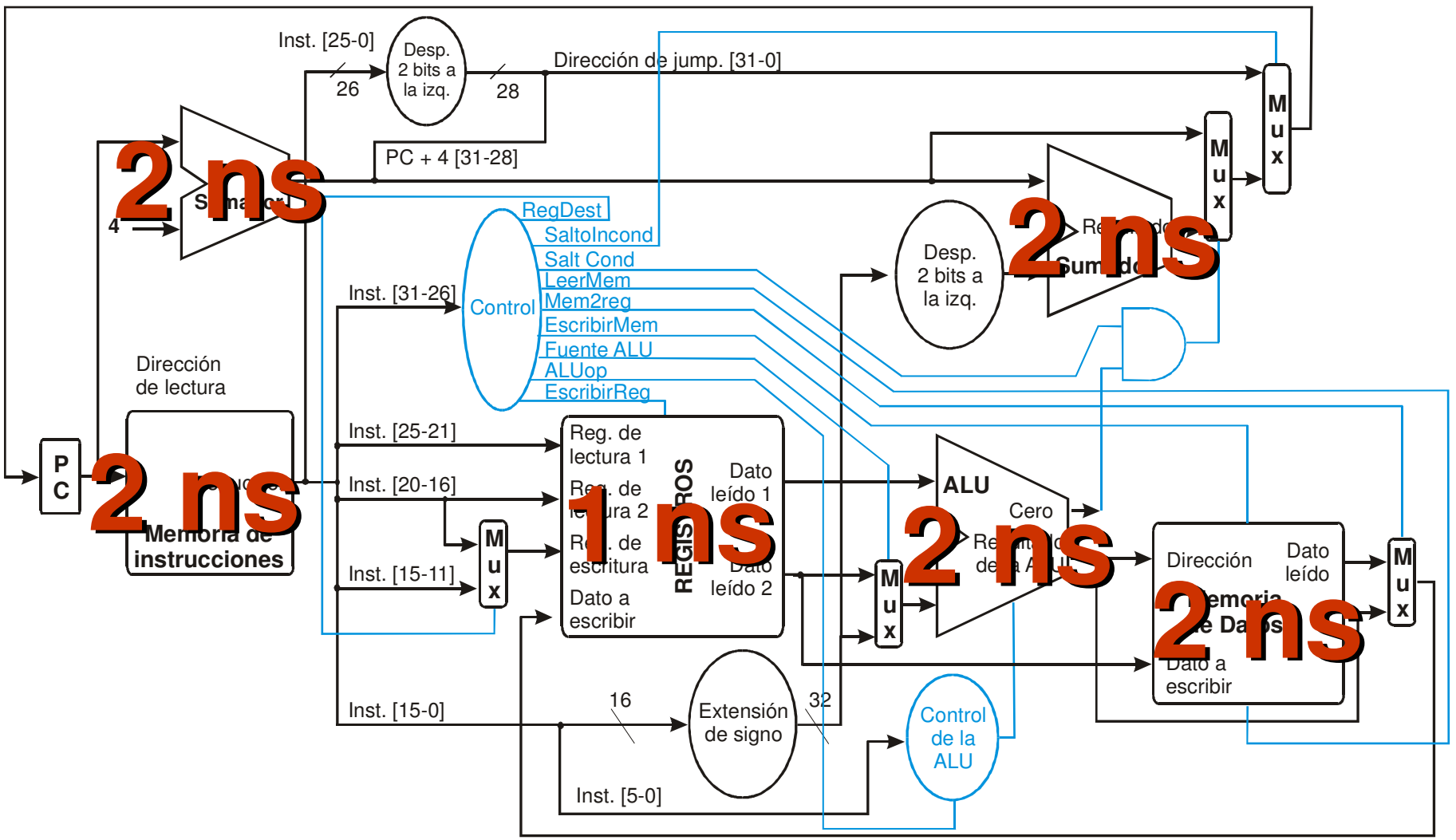
- El ciclo de reloj está definido por la instrucción más lenta
- No es posible reutilizar ninguna unidad funcional

- Estos inconvenientes se verían agravados en una arquitectura más compleja que la arquitectura MIPS

Evaluación del rendimiento

- Supóngase los tiempos de ejecución de las unidades funcionales siguientes:
  - Acceso a memoria: 2ns
  - ALU y sumadores: 2ns
  - Acceso a registros: 1ns
- ¿Cúal de las siguientes realizaciones será más rápida?
  - Una realización en la que cada instrucción se ejecuta en un ciclo de tamaño fijo (cada instrucción tarda lo que tardaría la más lenta).
  - Una realización donde cada instrucción se ejecuta en un ciclo de longitud variable (cada instrucción tarda únicamente lo necesario)

Cálculo del ciclo de reloj



Cálculo del ciclo de reloj

Tipo de instrucción	Unidades funcionales utilizadas por cada tipo de instrucción					ns
Aritmética	Cargar instrucción	Lectura de registros	ALU	Escritura de registros		**6**
lw	Cargar instrucción	Lectura de registros	ALU	Lectura memoria	Escritura de registros	**8**
sw	Cargar instrucción	Lectura de registros	ALU	Escritura en memoria		**7**
Salto condicional	Cargar instrucción	Lectura de registros	ALU			**5**
Jump	Cargar instrucción					**2**

Evaluación del rendimiento

Aritméticas: 44%	lw: 24%	sw: 12%
Saltos condicionales: 18%		Jump: 2%

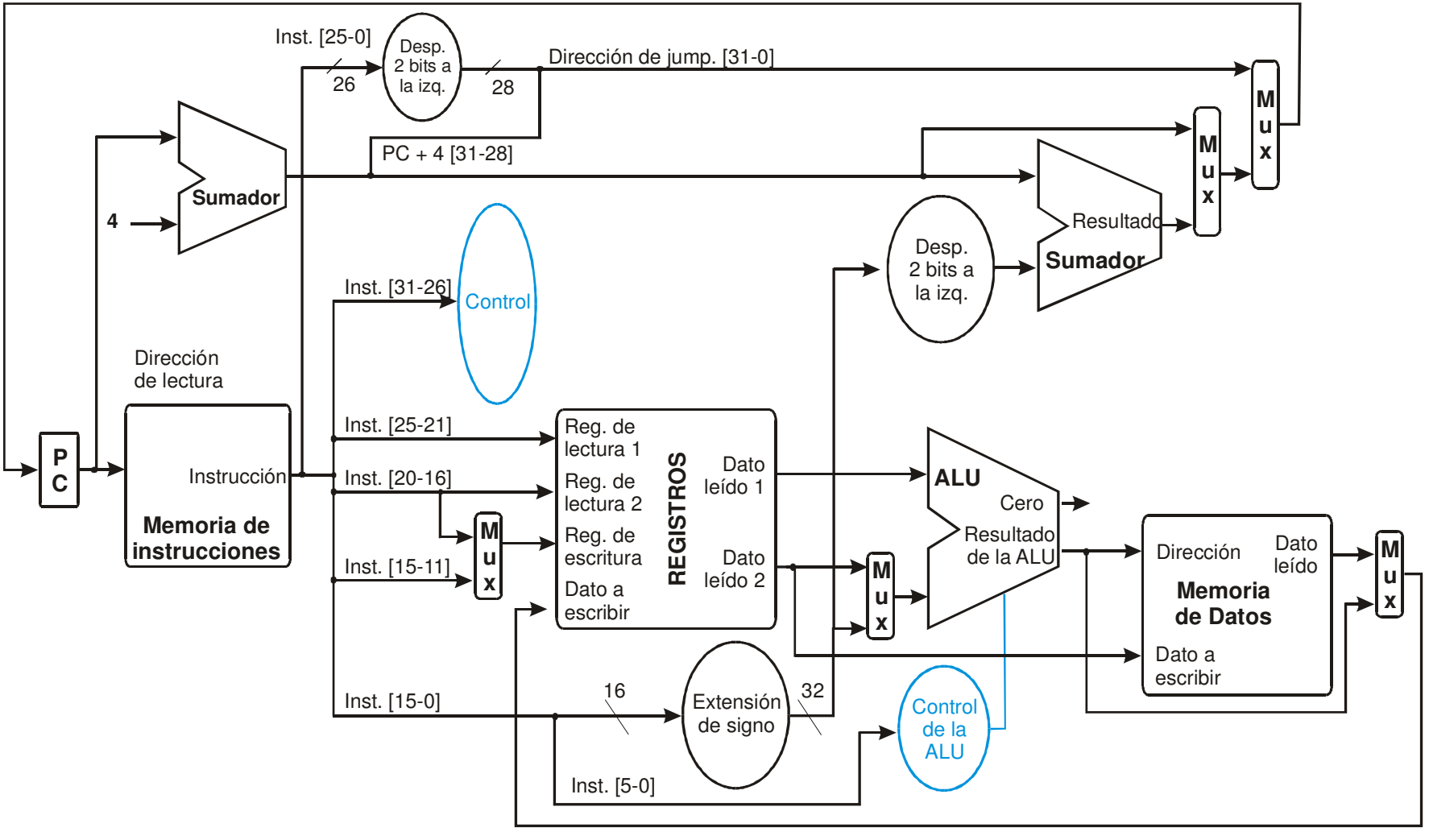
Tiempo medio ejecución para monociclo: 8 ns

Tiempo medio ejecución ideal:

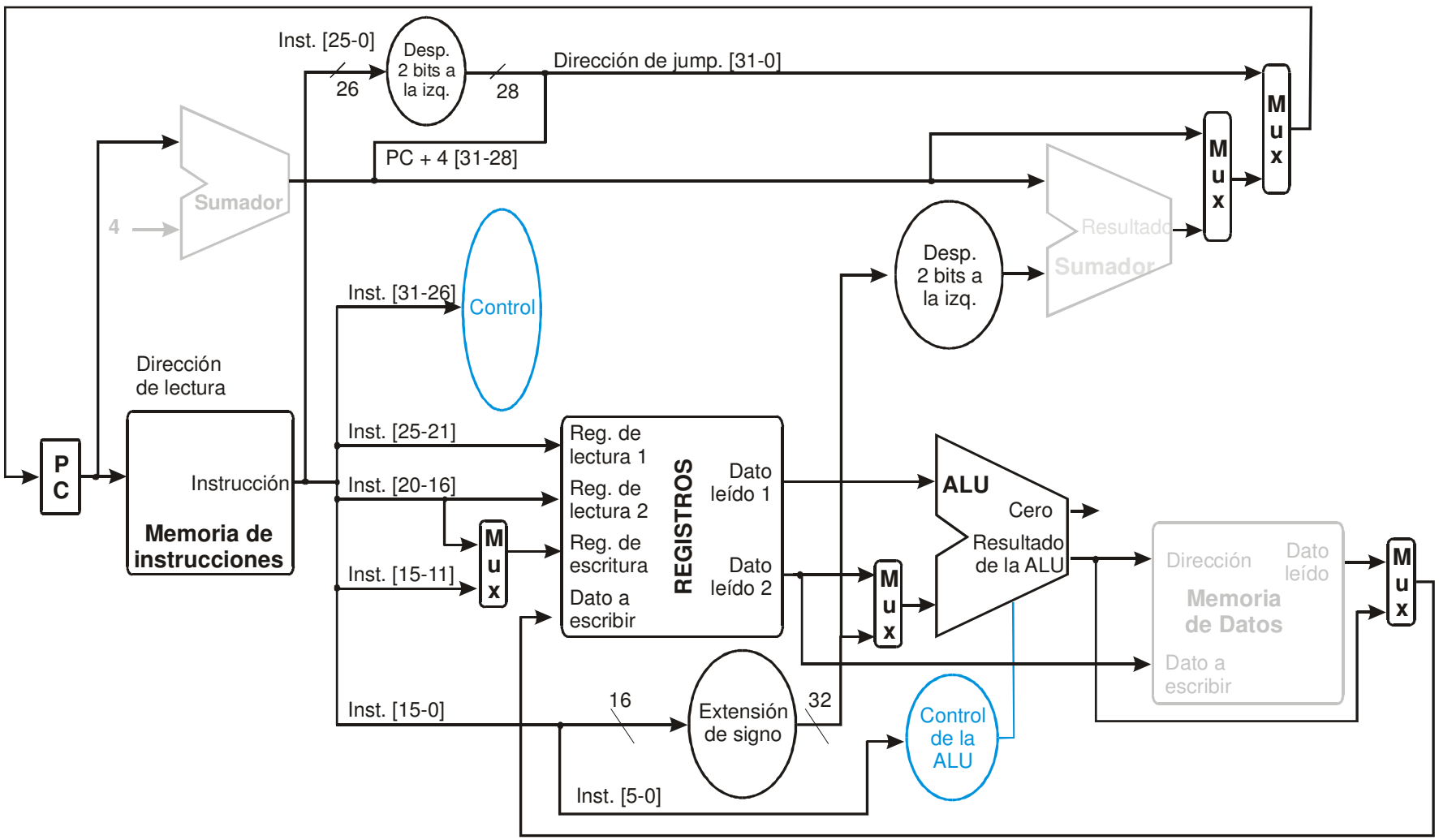
6 x 44% + 8 x 24% + 7 x 12% + 5 x 18% + 2 x 2% = **6.3 ns**

Rendimiento relativo: 8 / 6.3 = 1.27

Reducción de costes



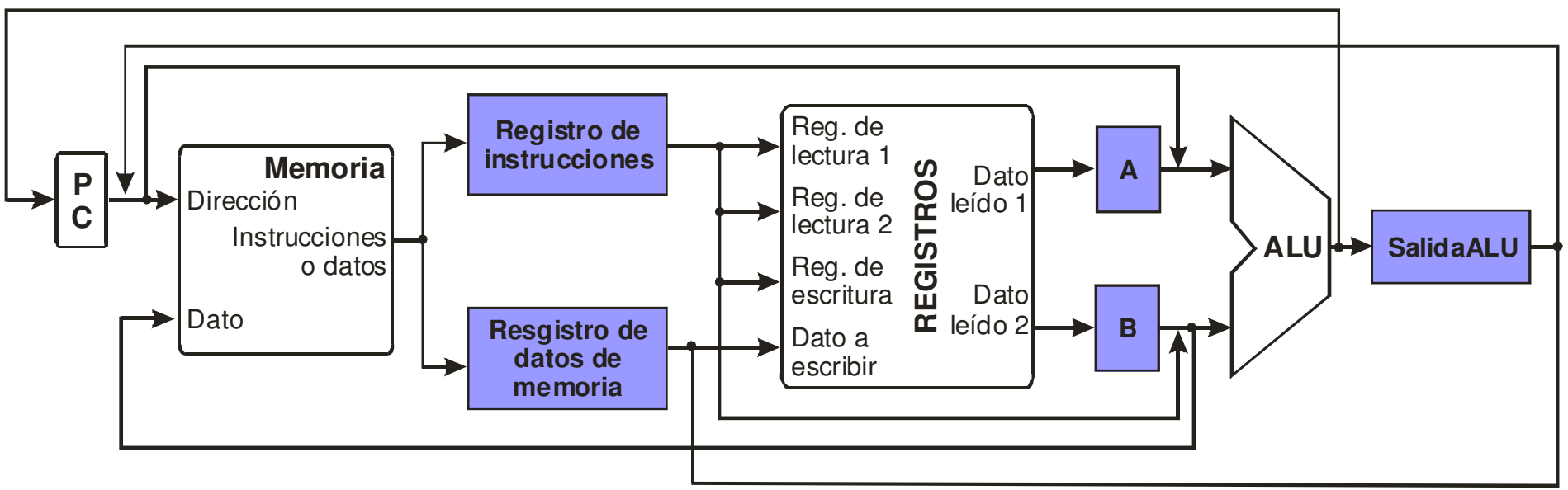
Reducción de costes



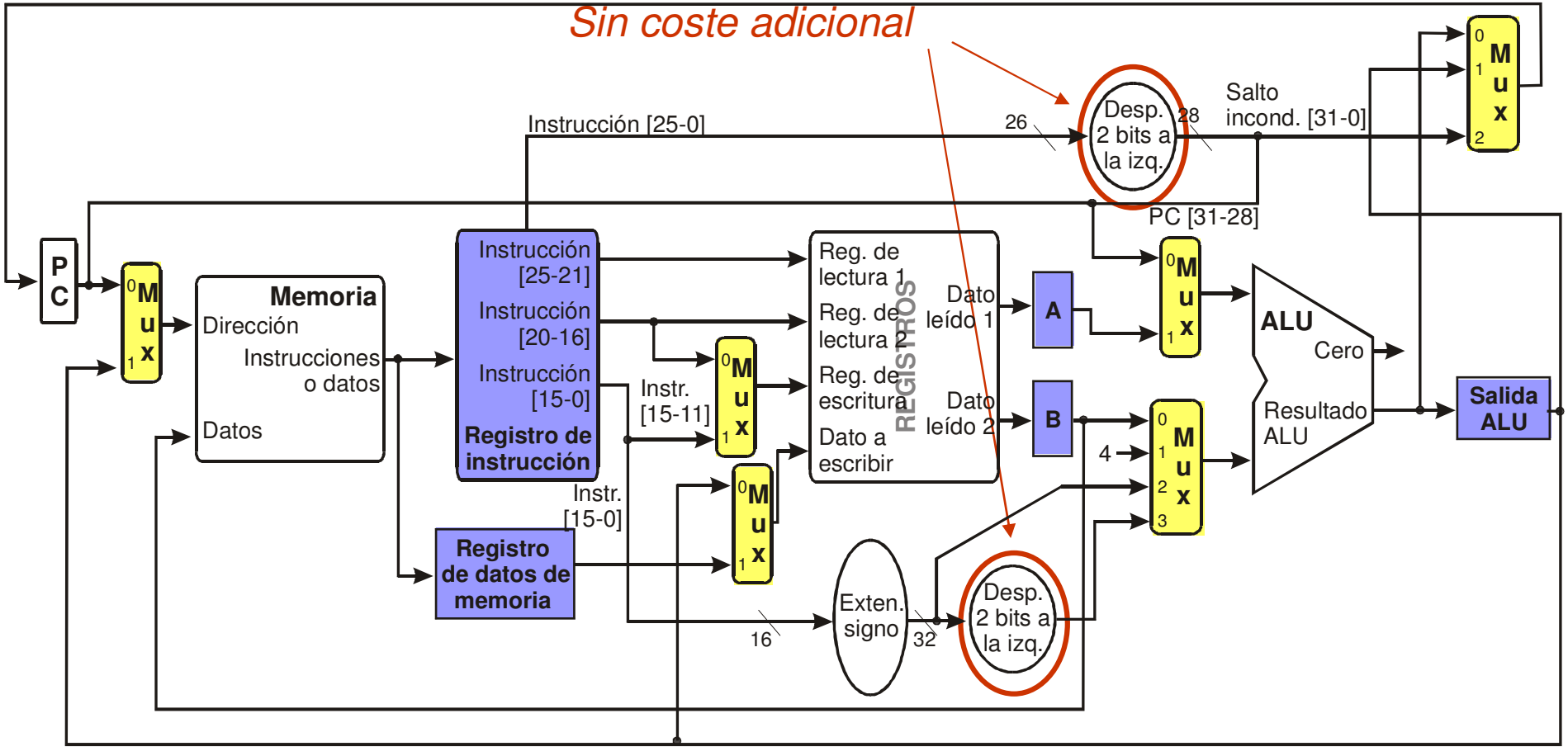
Implementación multiciclo

- Dedicaremos varios ciclos a cada instrucción
- Necesitaremos
  - Más multiplexores
  - Más registros
- Cada dato y resultado estará en un registro para que no se destruya al terminar el ciclo

Esquema multiciclo

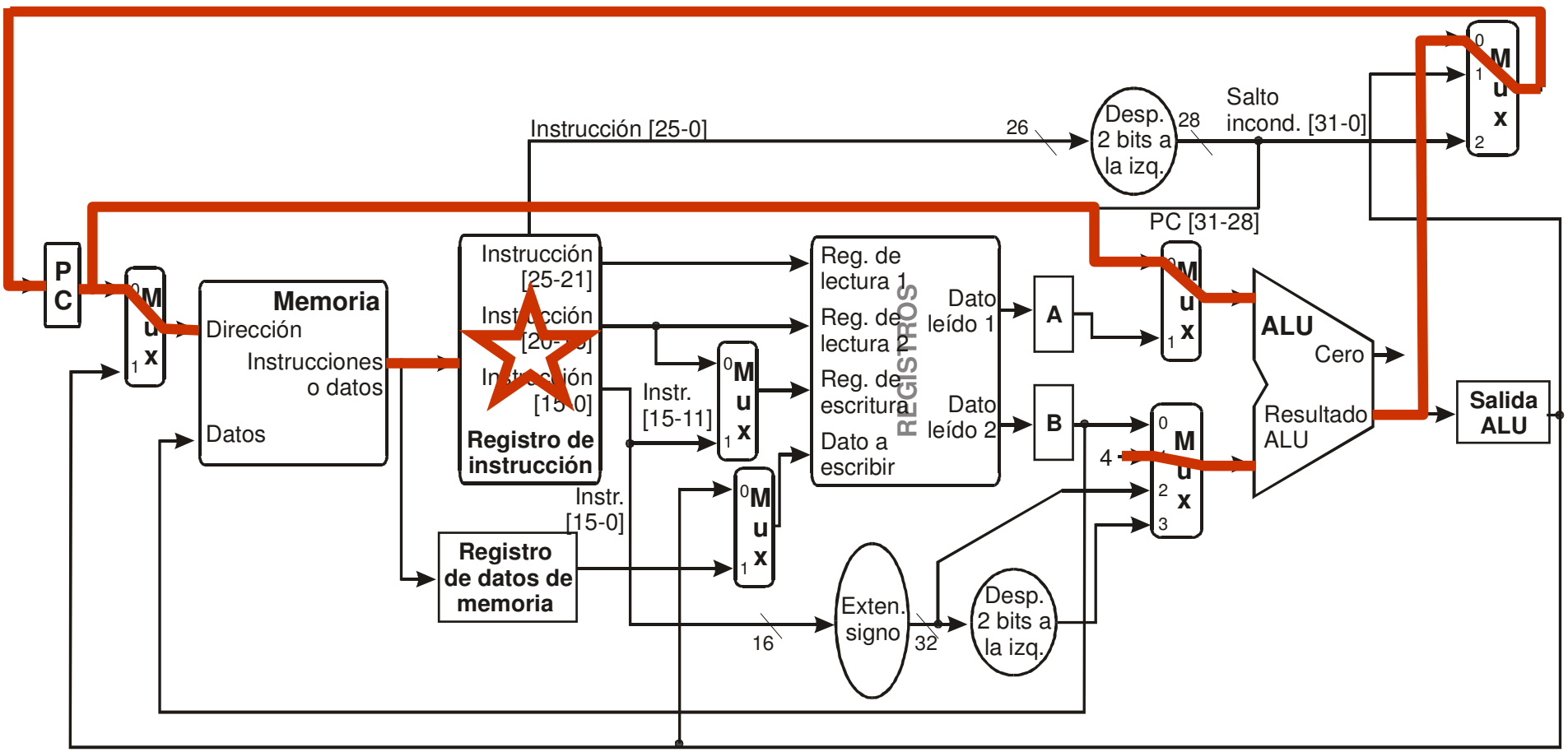


Implementación multiciclo



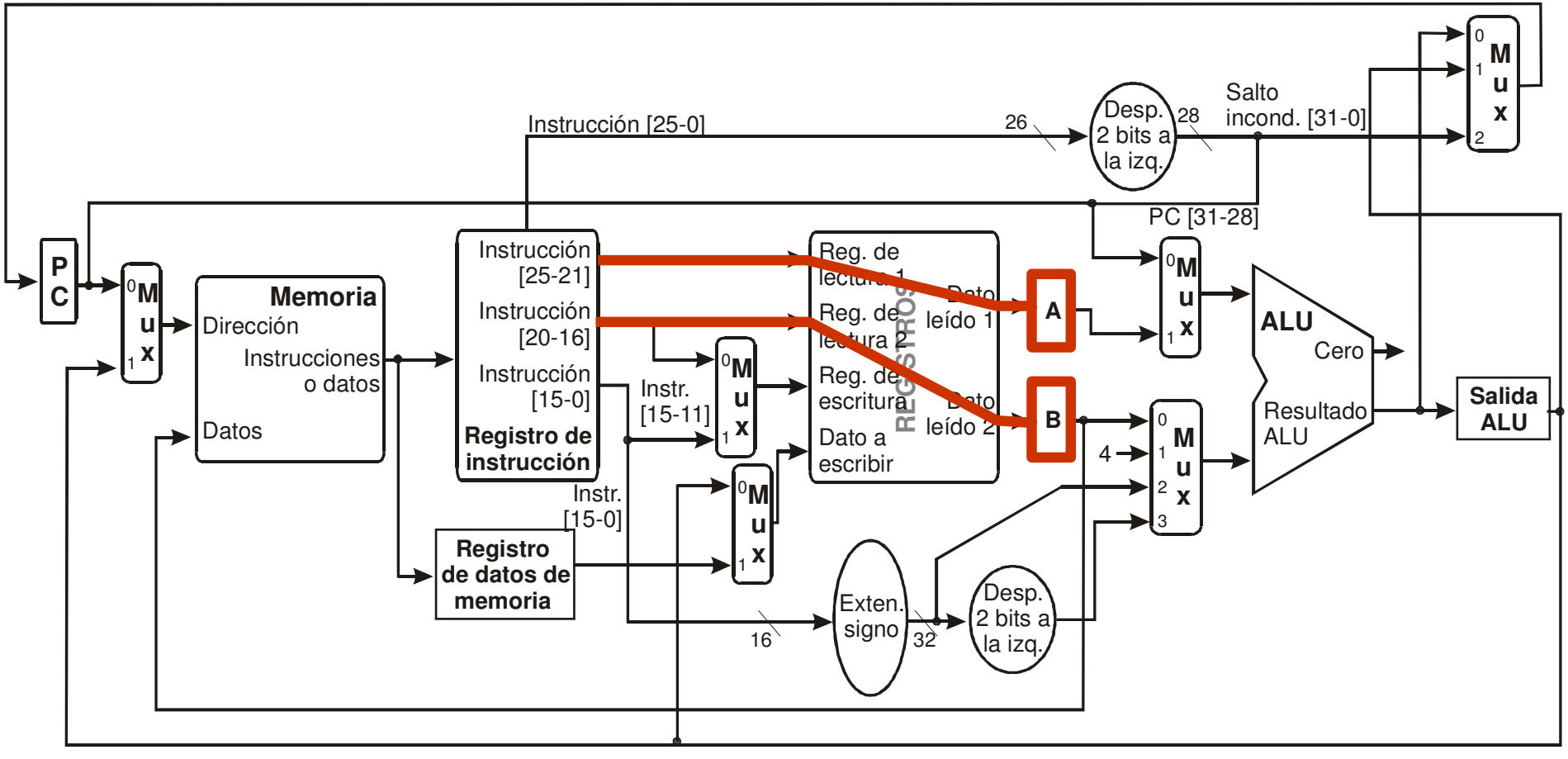
Carga de una instrucción

IR=Memoria[PC]
PC=PC+4



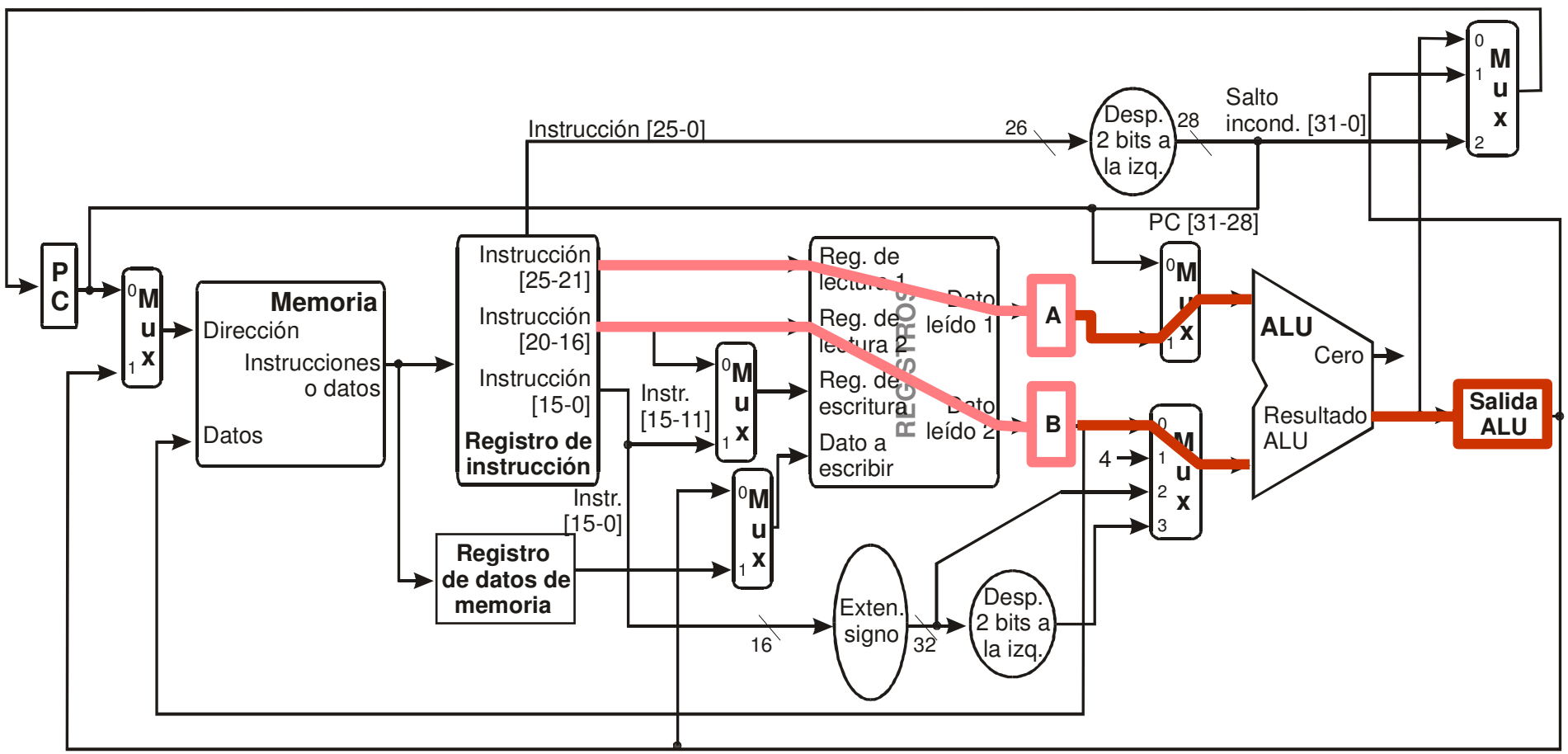
Instrucciones Aritmético-Lógicas: Búsqueda de registros

A=Reg[IR[25-21]]
B=Reg[IR[20-16]]



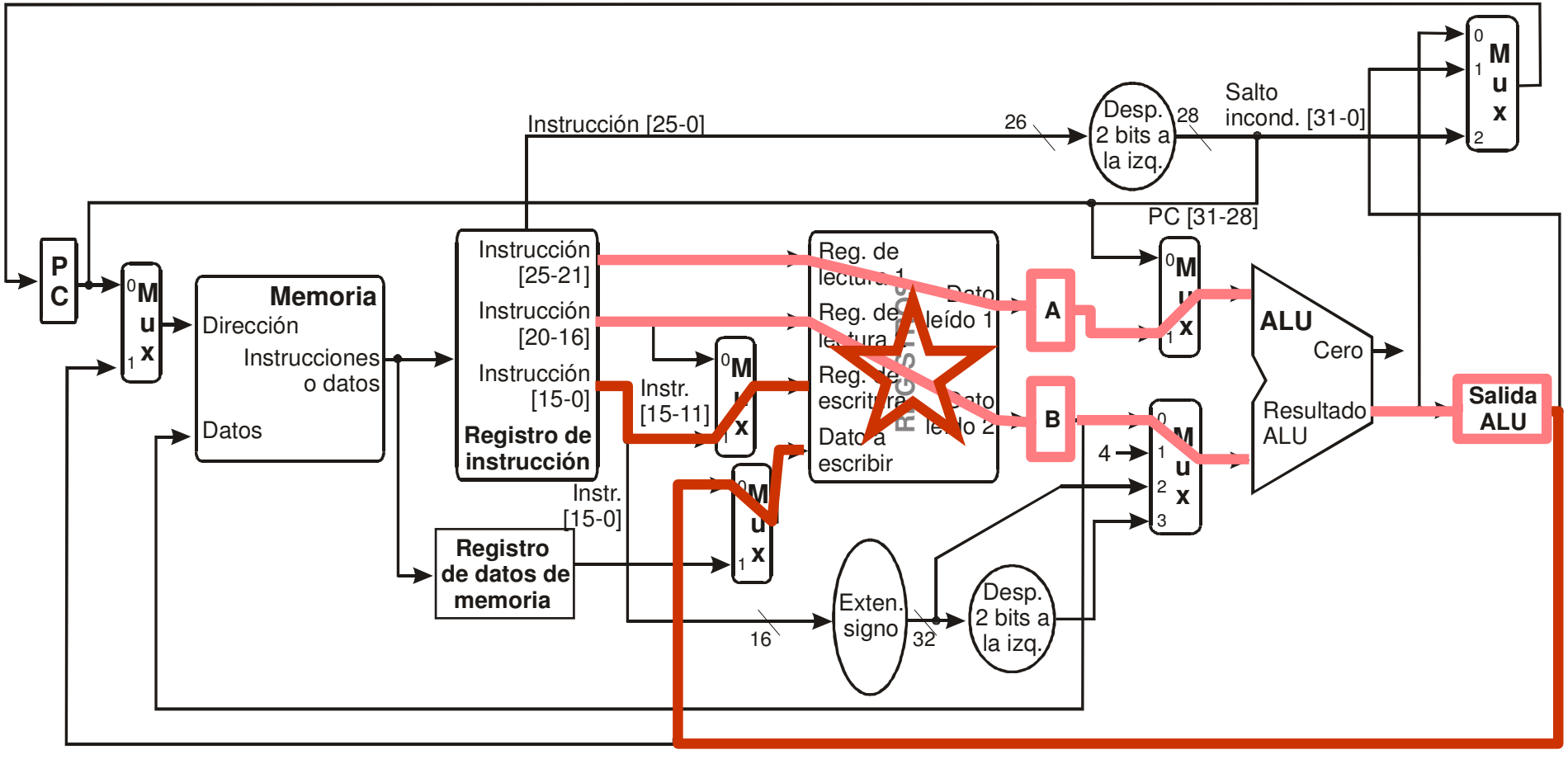
Instrucciones Aritmético-Lógicas: Ejecución

ALUOut = A op B



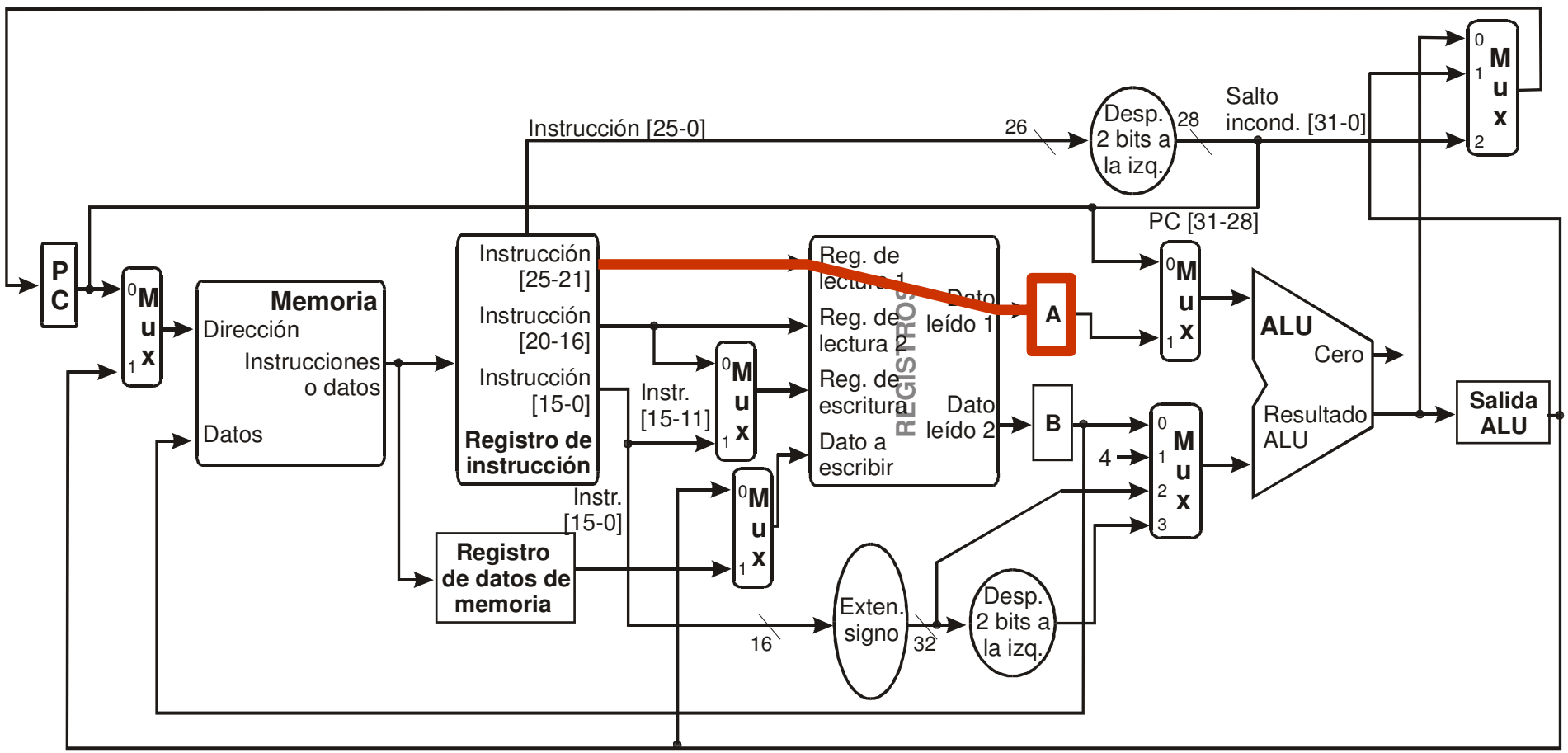
Instrucciones Aritmético-Lógicas: Guarda resultados en registro

Reg[IR[15-11]]=ALUOut



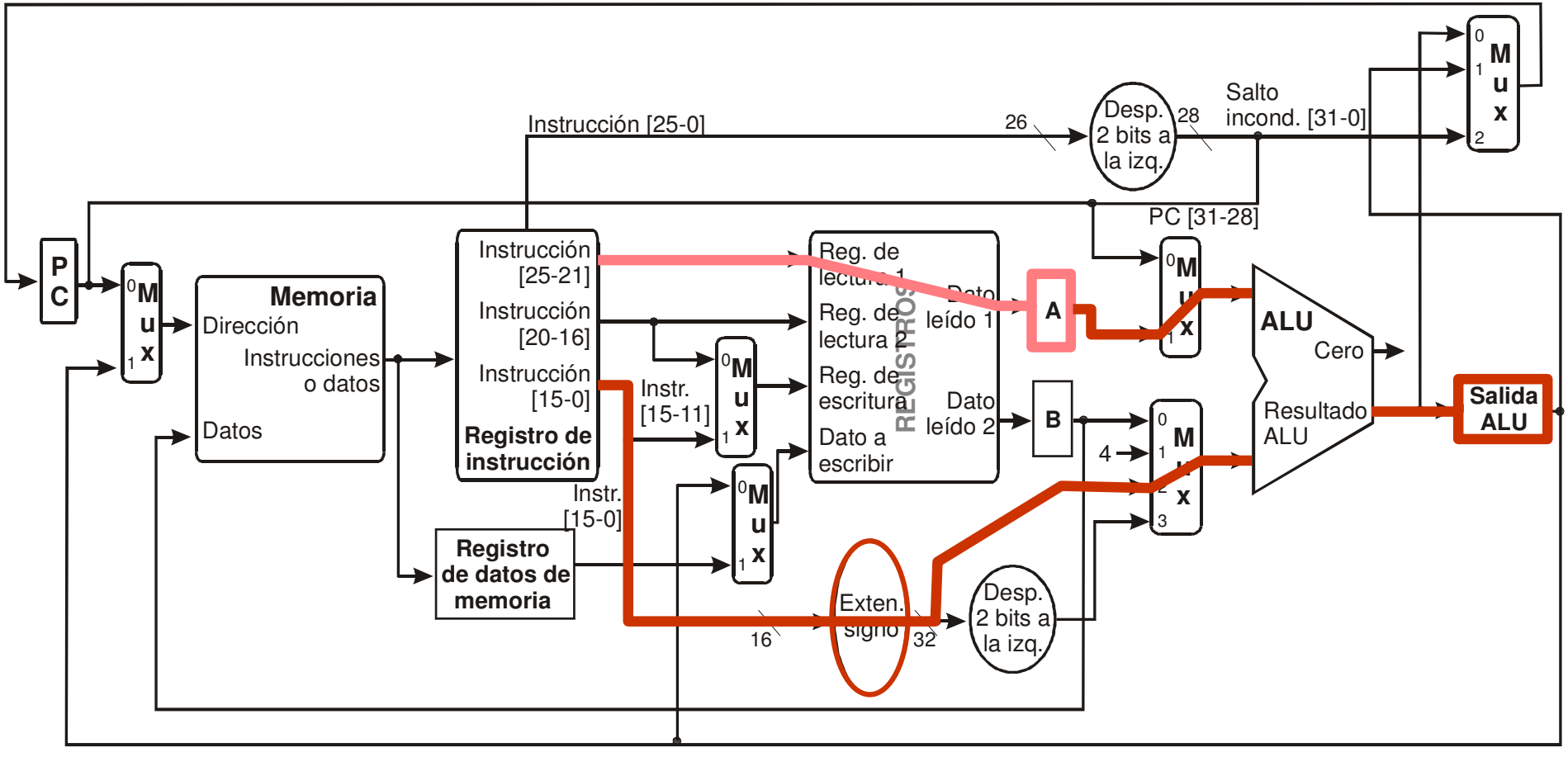
Instrucción lw: Búsqueda de registros

A=Reg[IR[25-21]]
B=Reg[IR[20-16]]



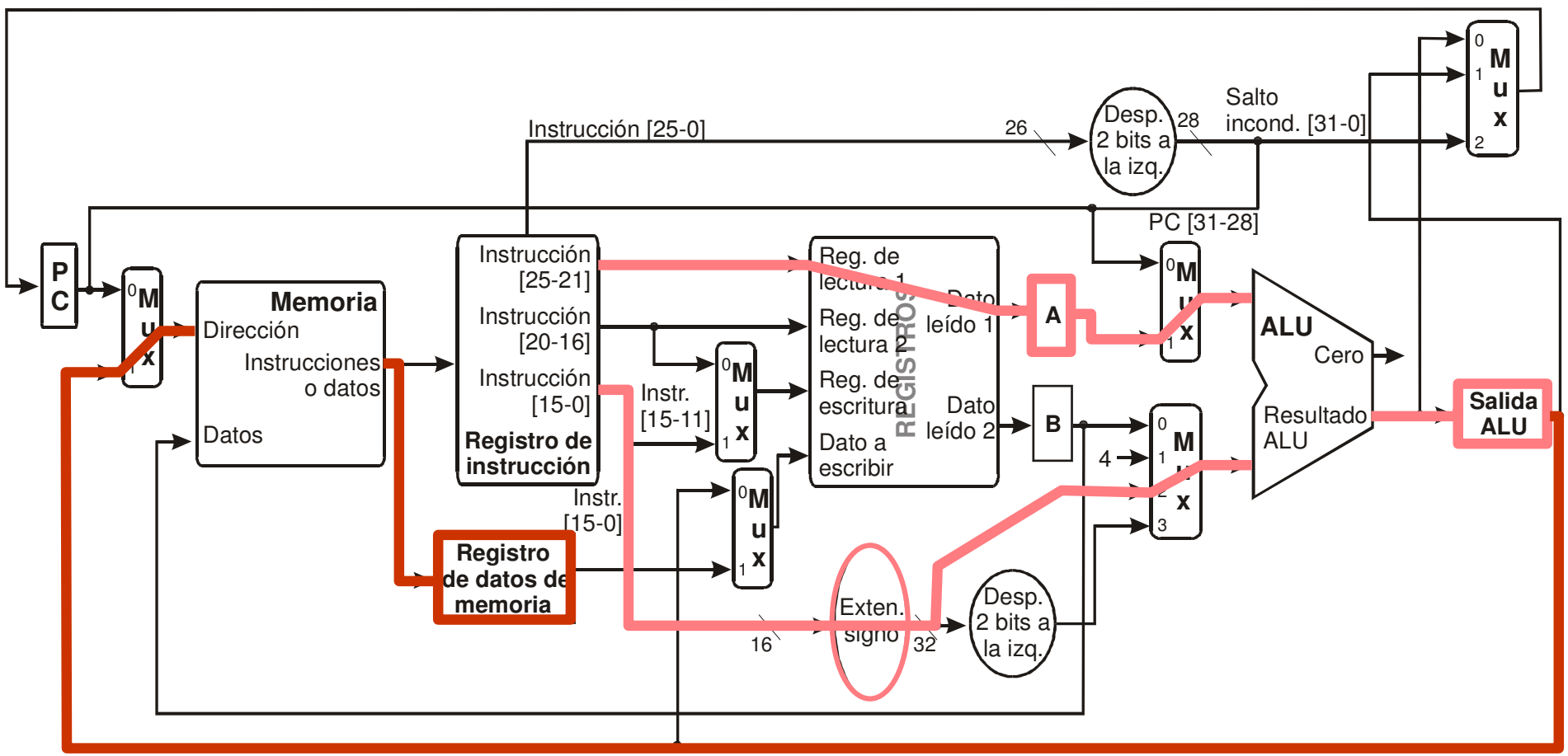
Instrucción lw:
Cálculo de la dirección

ALUOut = A + signo extendido (IR[15-0])



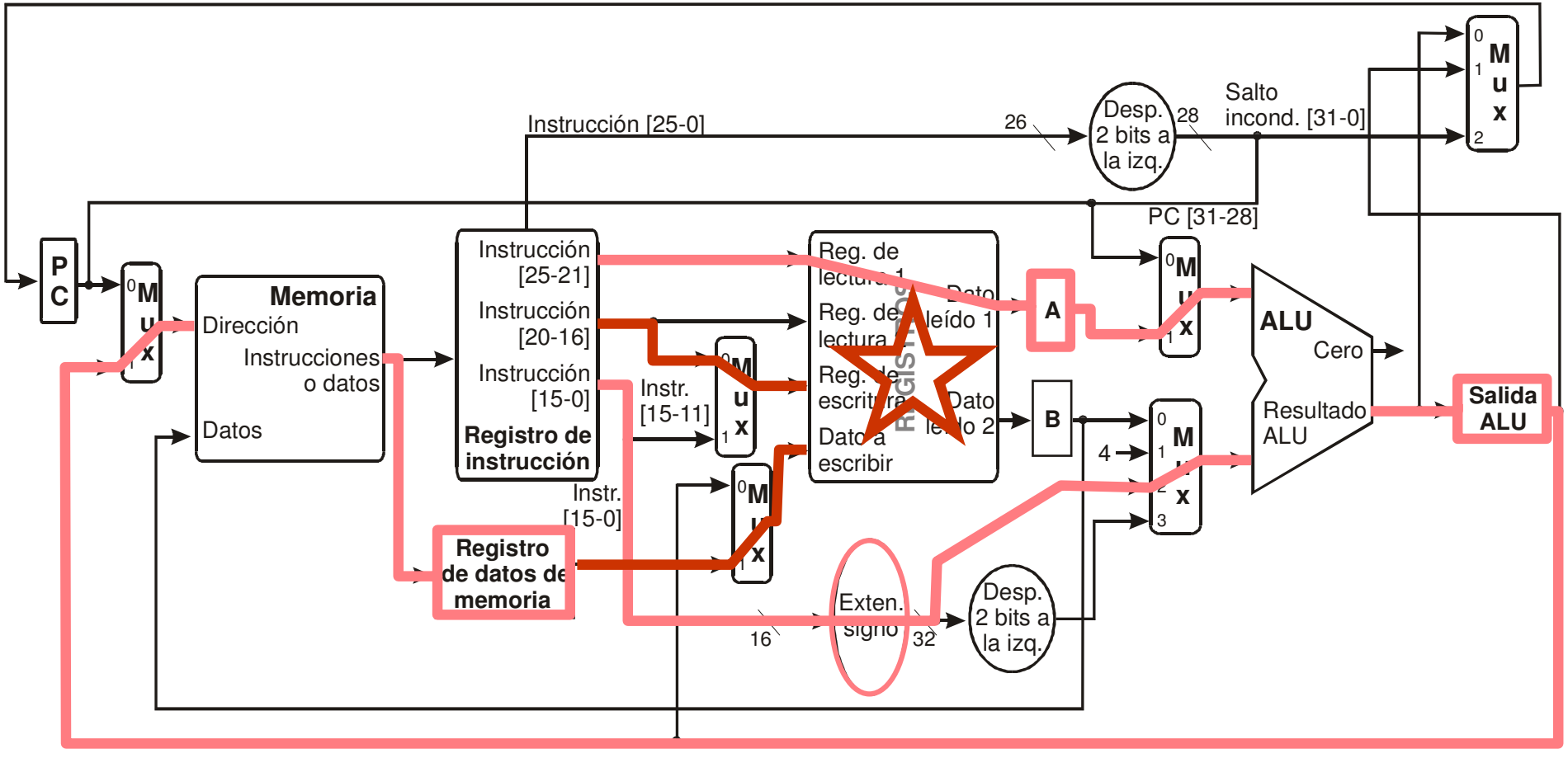
Instrucción lw: Lectura de memoria

MDR=Memoria[ALUOut]



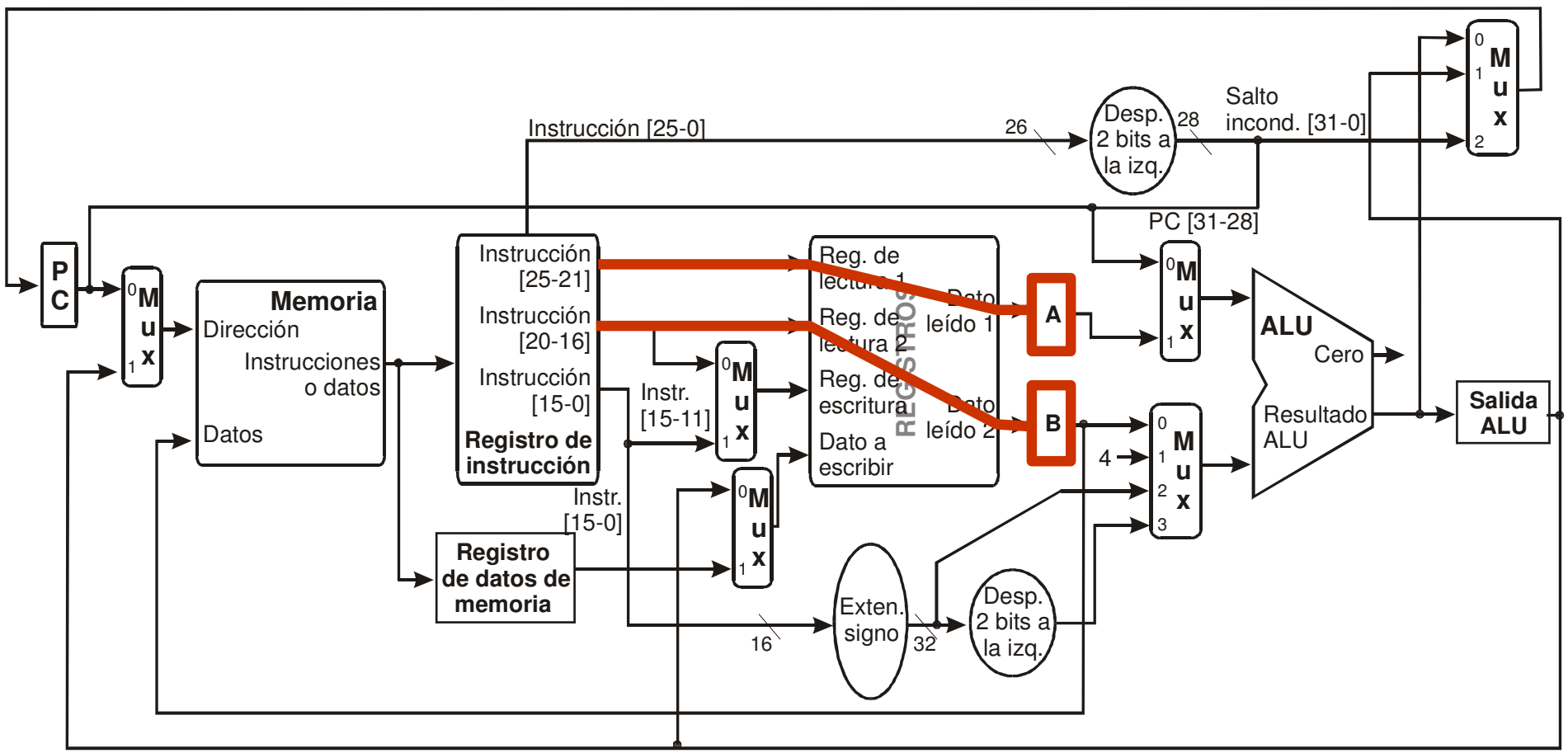
Instrucción lw:
Guarda dato en registro

Reg[IR[20-16]] = MDR



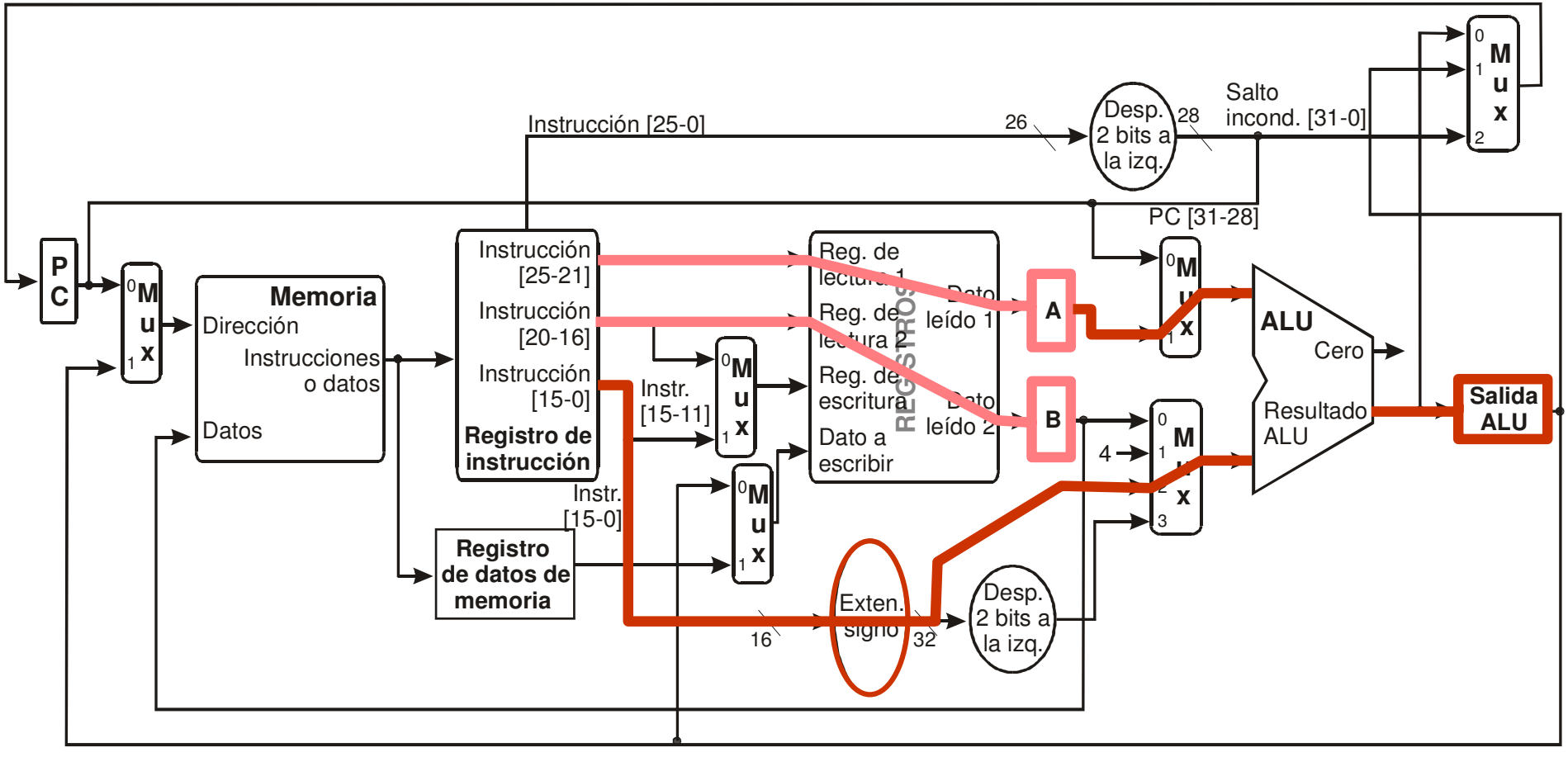
Instrucción sw: Búsqueda de registros

A=Reg[IR[25-21]]
B=Reg[IR[20-16]]



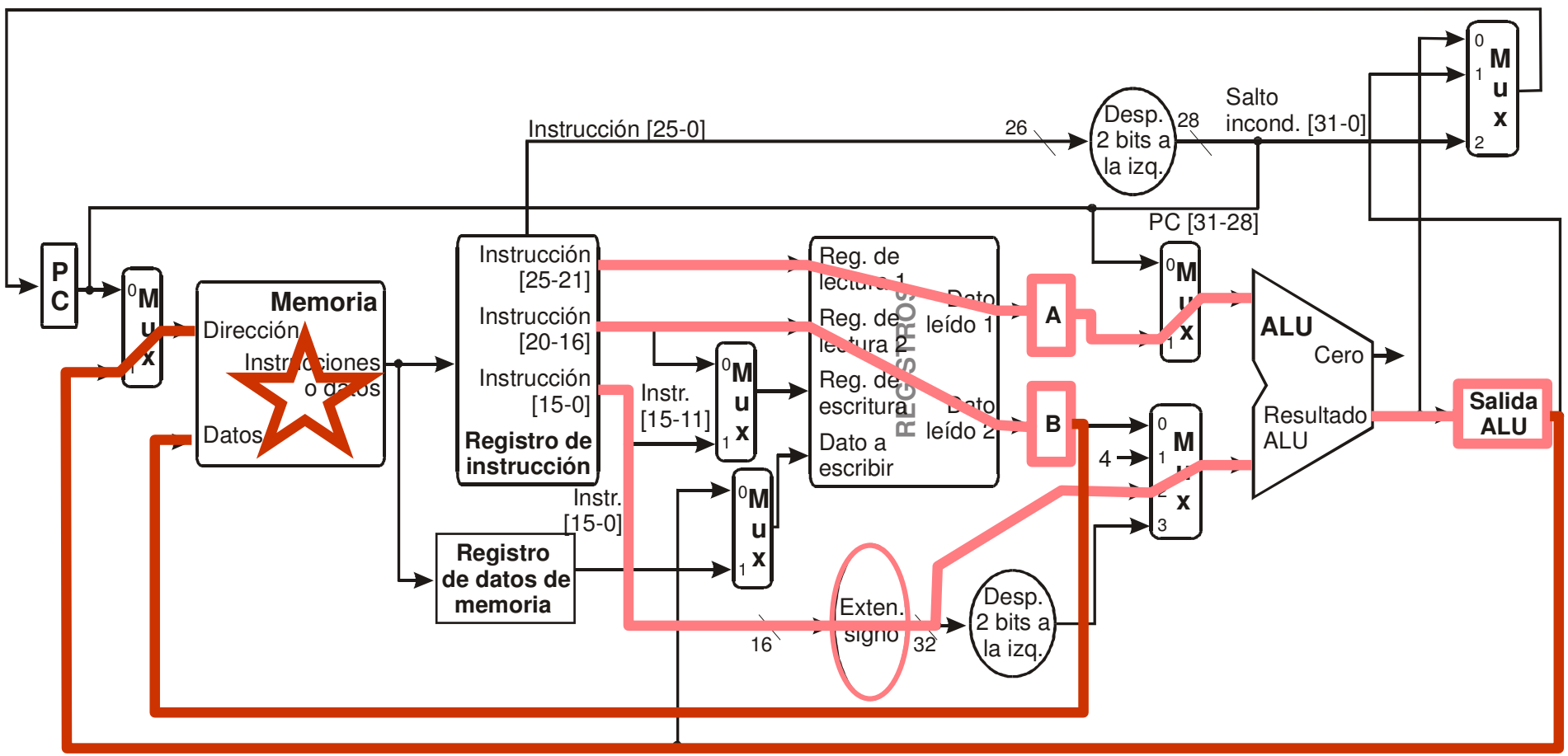
Instrucción sw:
Cálculo de la dirección

ALUOut = A + signo extendido (IR[15-0])



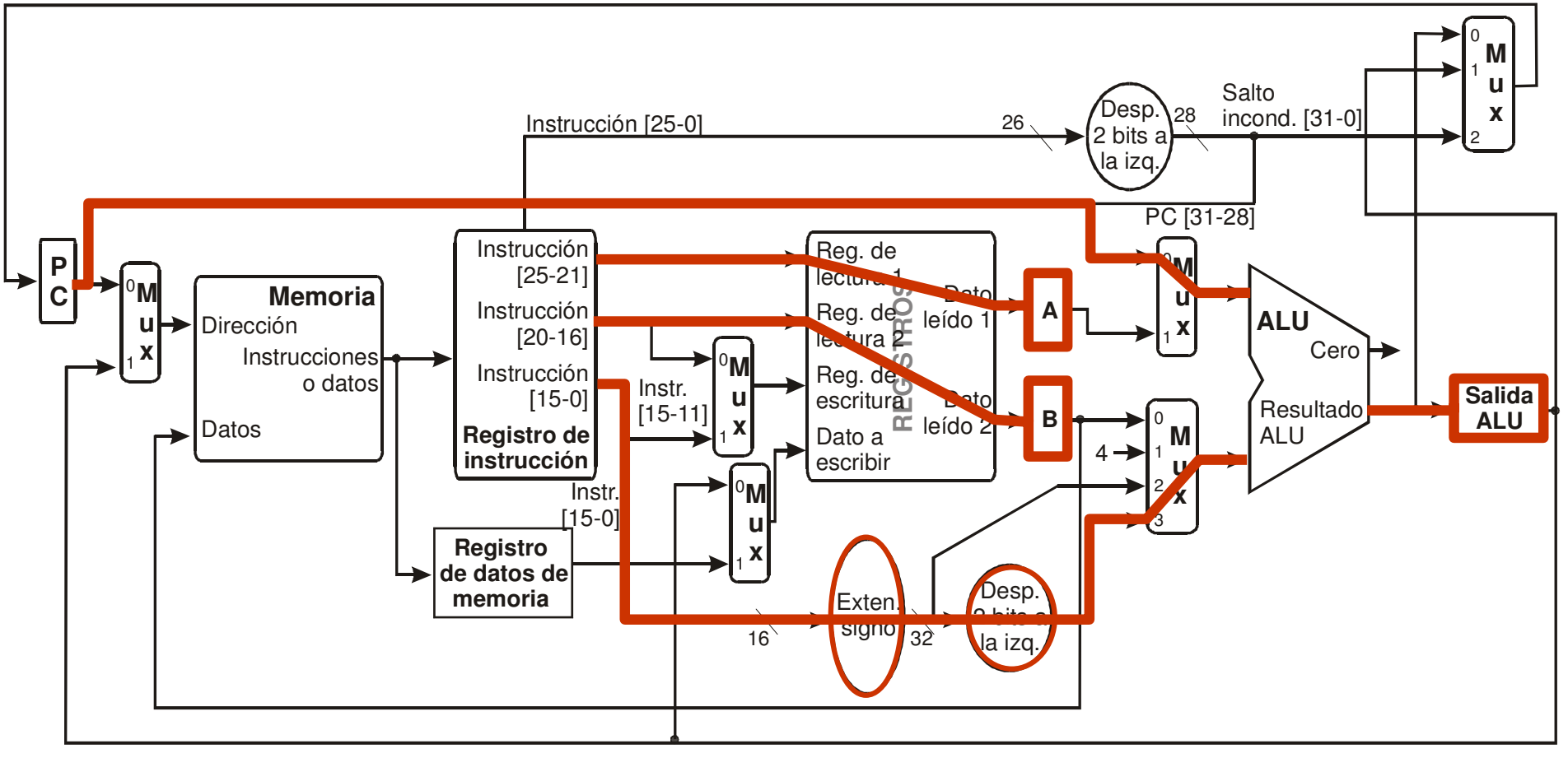
Instrucción sw: Escritura en memoria

Memoria[ALUOut] = B



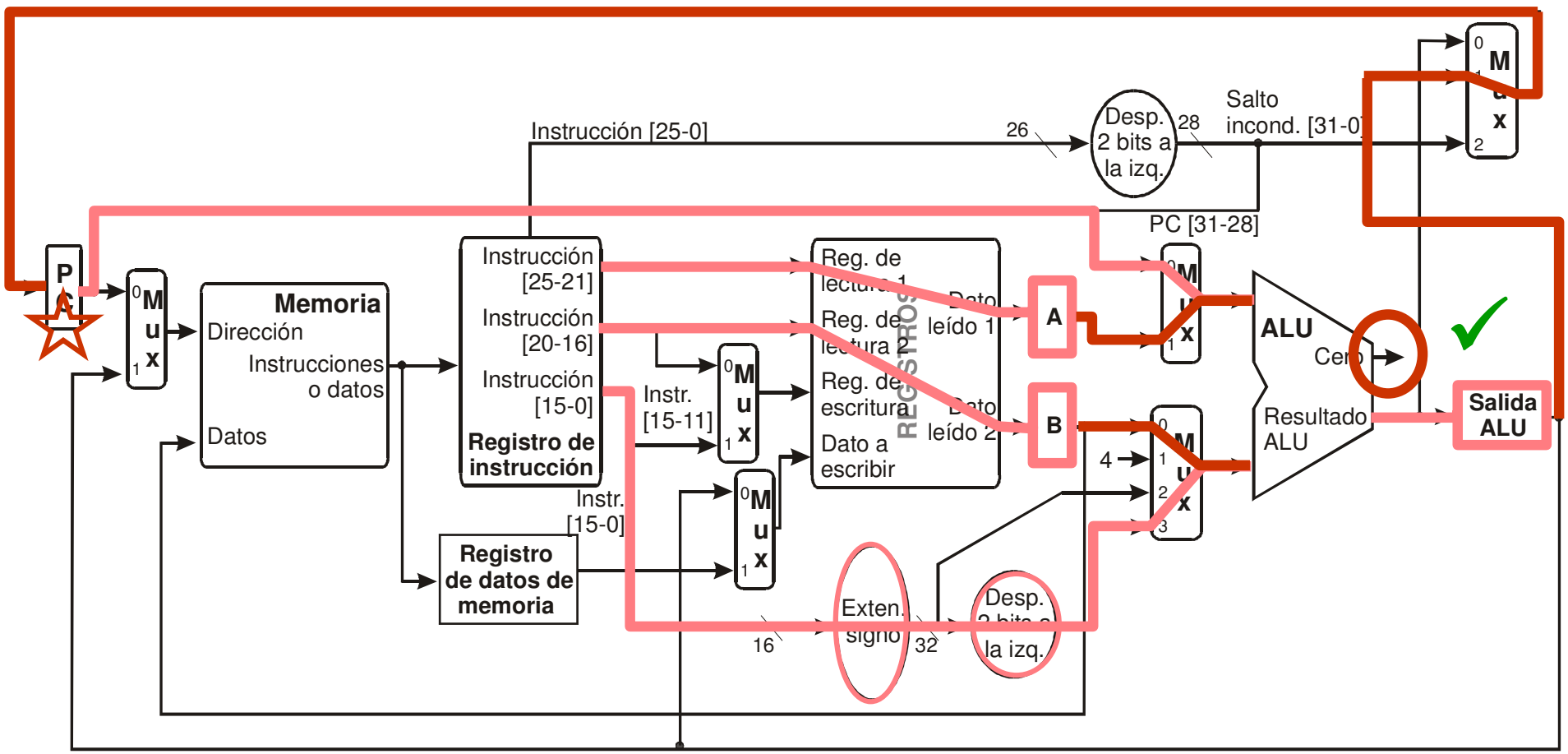
Instrucción beq: Búsqueda de registros

A=Reg[IR[25-21]]
B=Reg[IR[20-16]]
ALUOut = PC + (signo extendido (IR[15-0]) << 2)

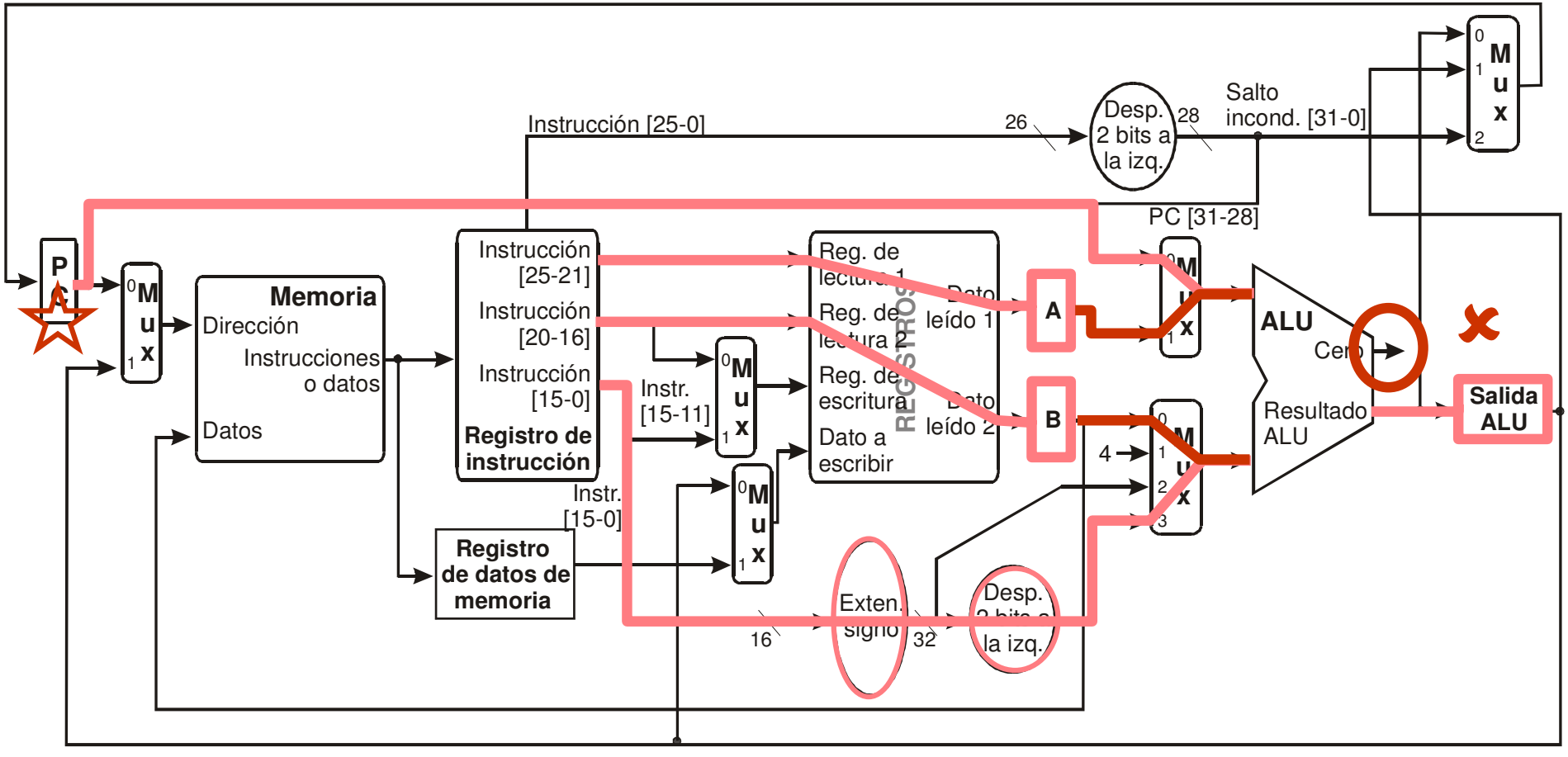


Instrucción beq: Comprobación con éxito

If (A==B) PC = ALUOut

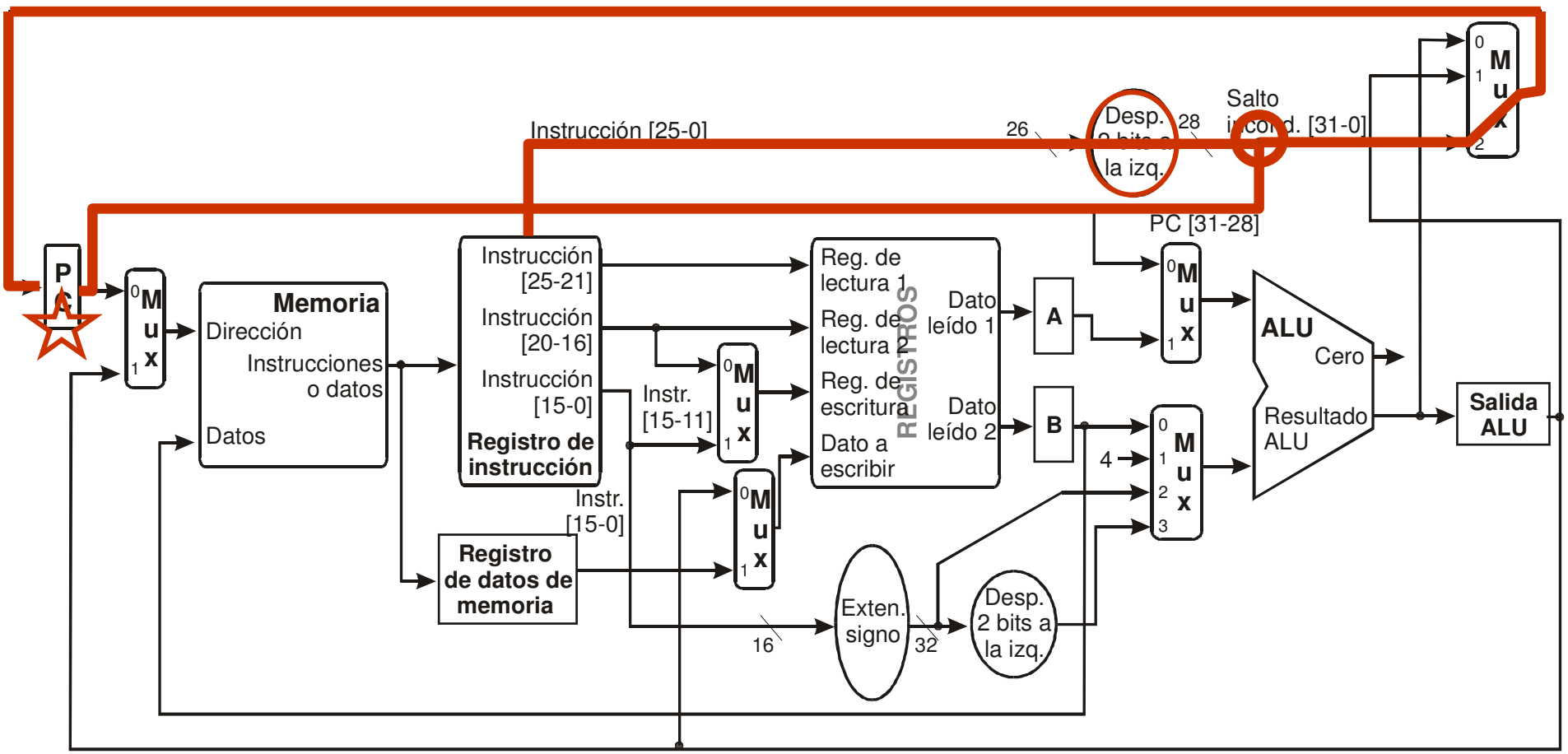


Instrucción beq: Comprobación sin éxito



Instrucción Jump: Ejecución del salto

PC = PC[31-28] || (IR[25-0] <<2)



CPI multiciclo

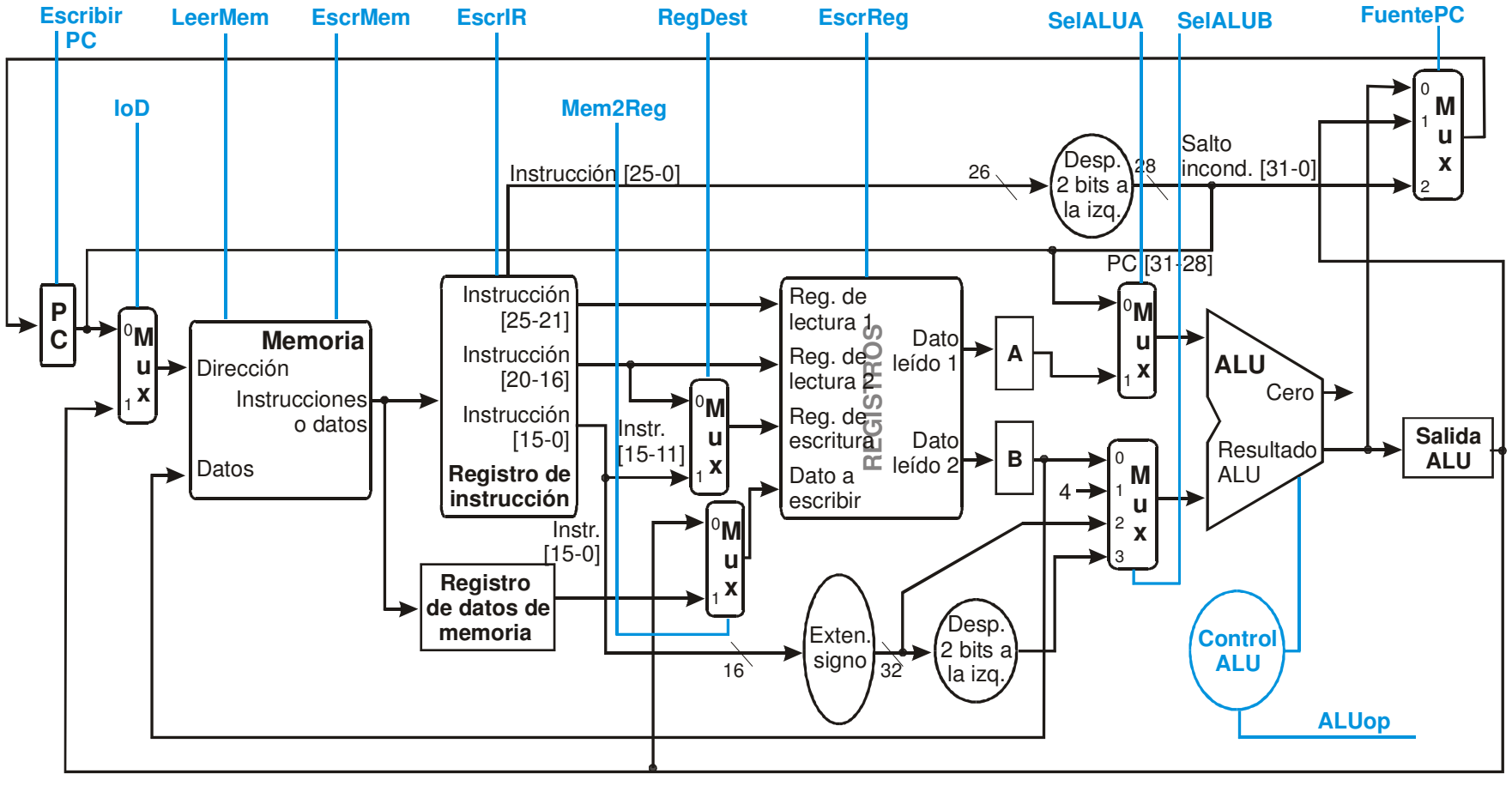
- Hemos repartido las operaciones de forma que el tiempo máximo de computación es el de una unidad funcional
- La unidades más lentas necesitan 2 ns para completar su funcionamiento
- Por tanto el ciclo de reloj durará 2 ns
- Por ejemplo, una operación aritmética necesita 4 ciclos = 8 ns

CPI multiciclo

Tipo de instrucción	%	Multiciclo
Aritmética	44	4
lw	24	5
sw	12	4
Salto condicional	18	3
Jump	2	3
		4,04

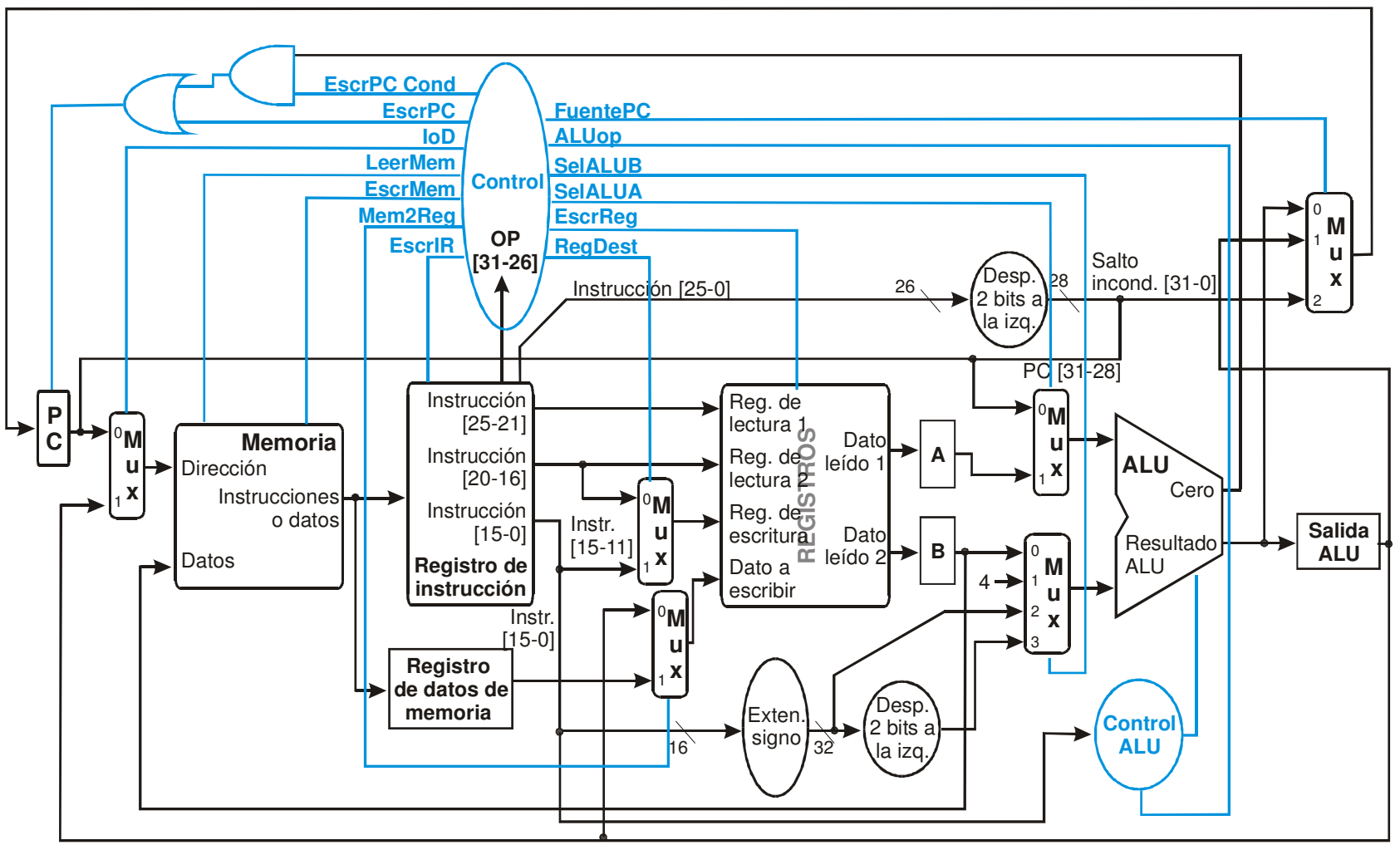
Diseño de la unidad de control

- Realización multiciclo: señales de control



Diseño de la unidad de control

- Realización multiciclo: señales de control



Diseño de la unidad de control

- El control del camino de datos multiciclo debe especificar:
  - √ Las señales que se van a inicializar en cada paso
  - √ El paso siguiente de la secuencia

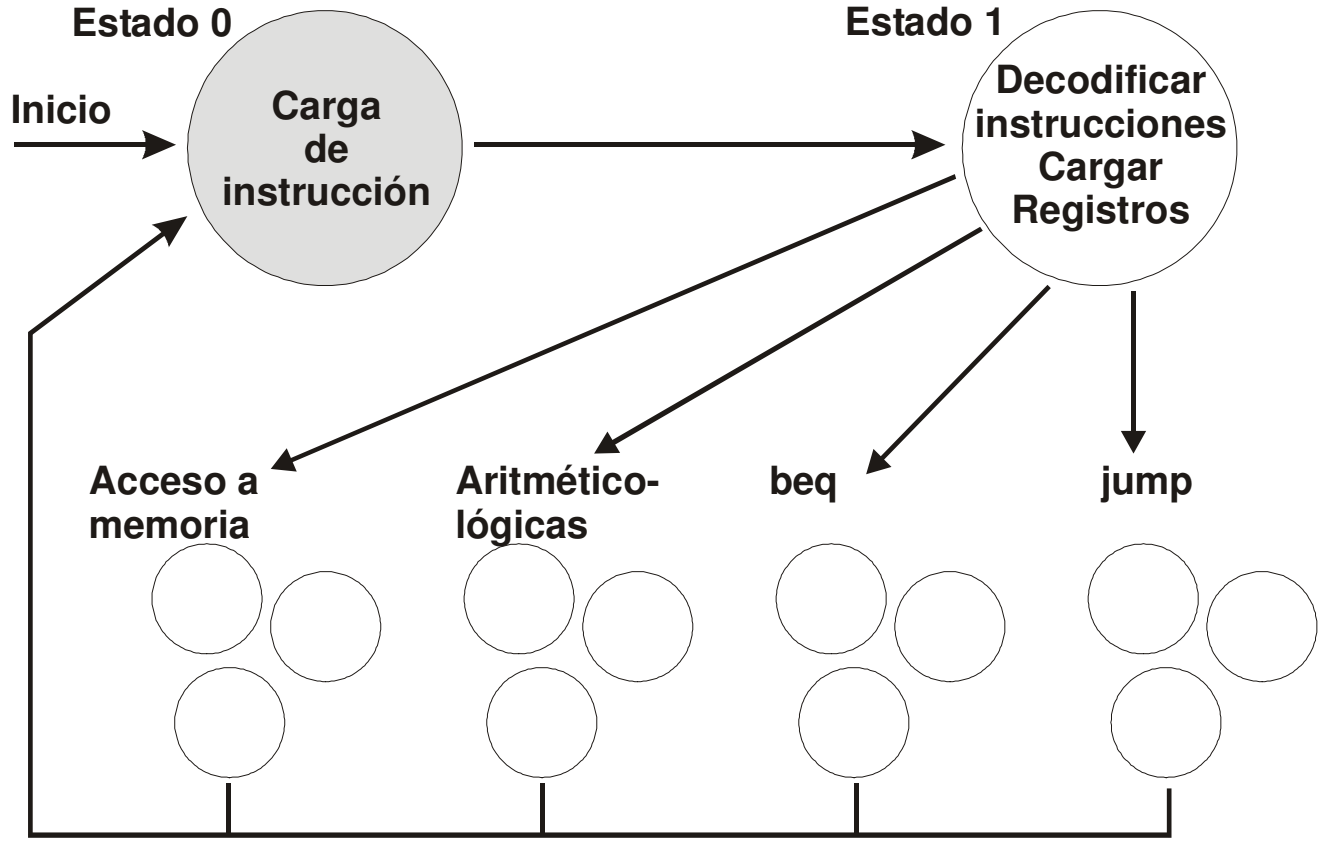
- Dos técnicas diferentes:
  - √ **Control cableado**. Se basa en las máquinas de estados finitos.
  - √ **Control microprogramado**. Se representa en forma de programa de control

Control cableado

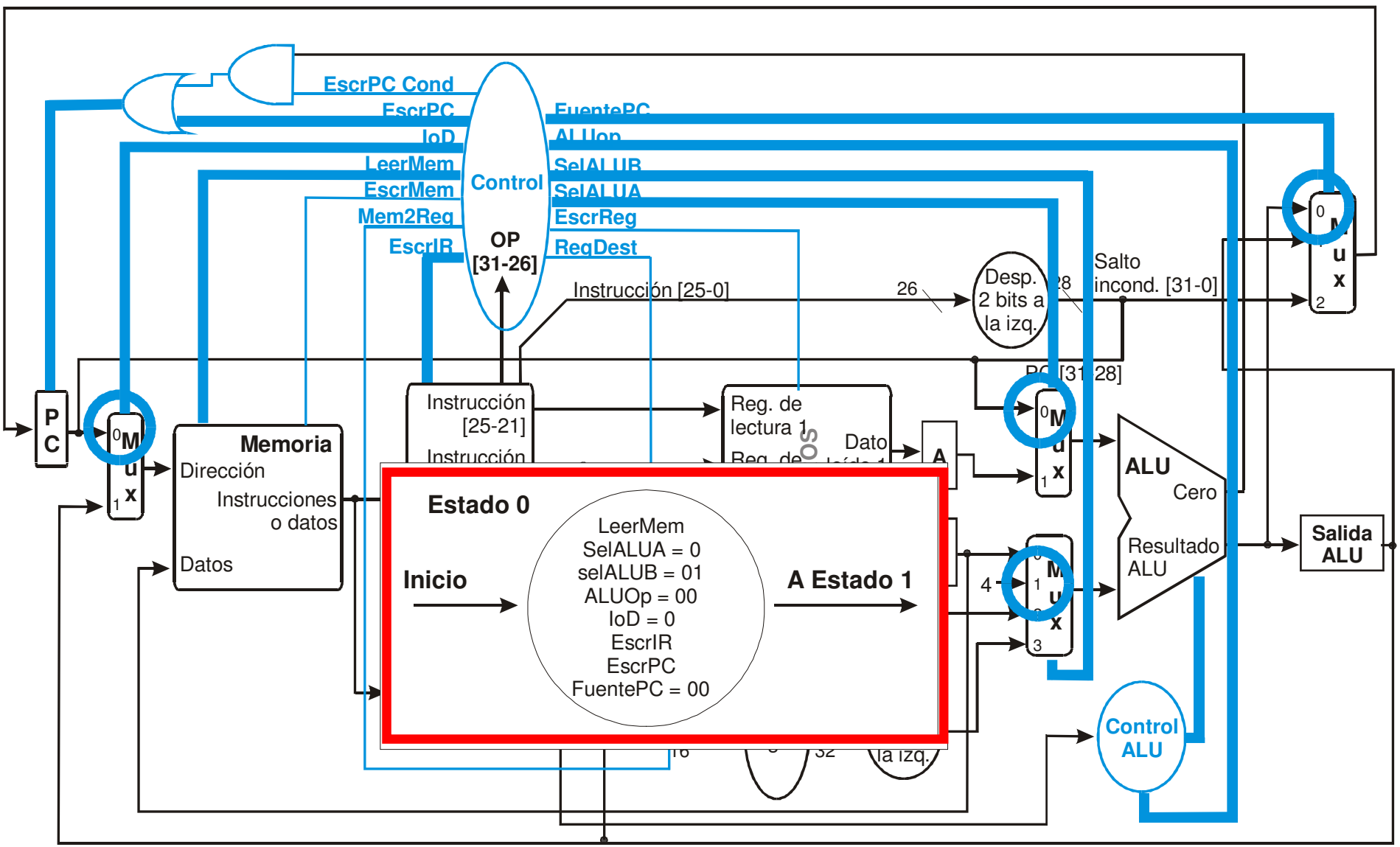
- Construiremos una máquina de estados finitos (autómata de Moore)
- El camino de datos multiciclo se controla con las salidas de la unidad de control (la máquina de estados)
- Las entradas de la unidad de control serán
  - √ Los bits de la instrucción
  - √ Los indicadores internos
- Máquina de estados finitos:
  - √ Cada estado de la máquina representa una etapa y tarda un ciclo de reloj
  - √ Los dos primeros pasos son idénticos para todas las instrucciones
  - √ A partir de la tercera etapa depende del código de operación
  - √ Después de la última etapa la máquina debe volver al estado inicial

Control cableado

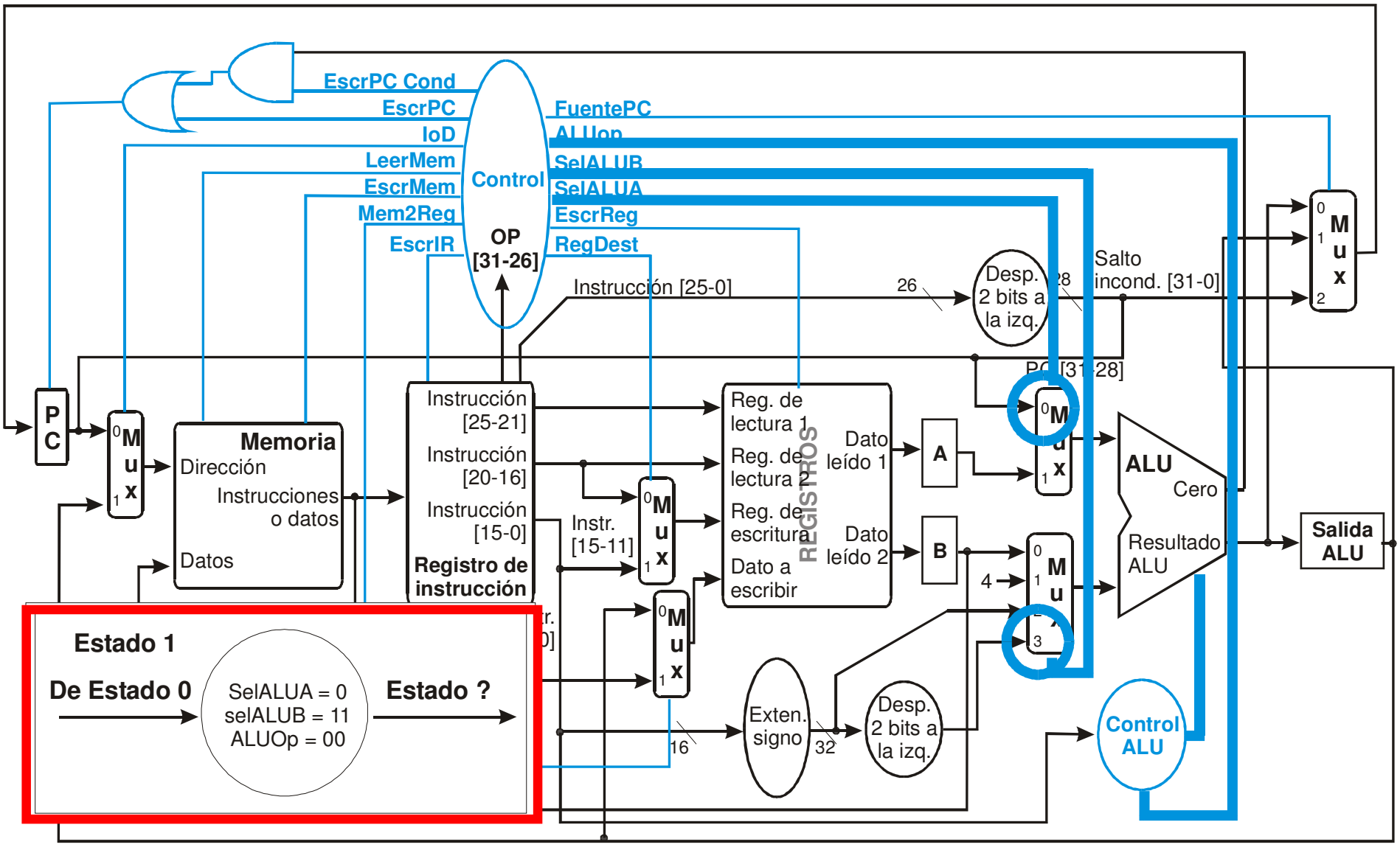
- Realización de máquinas de estados finitos de control:



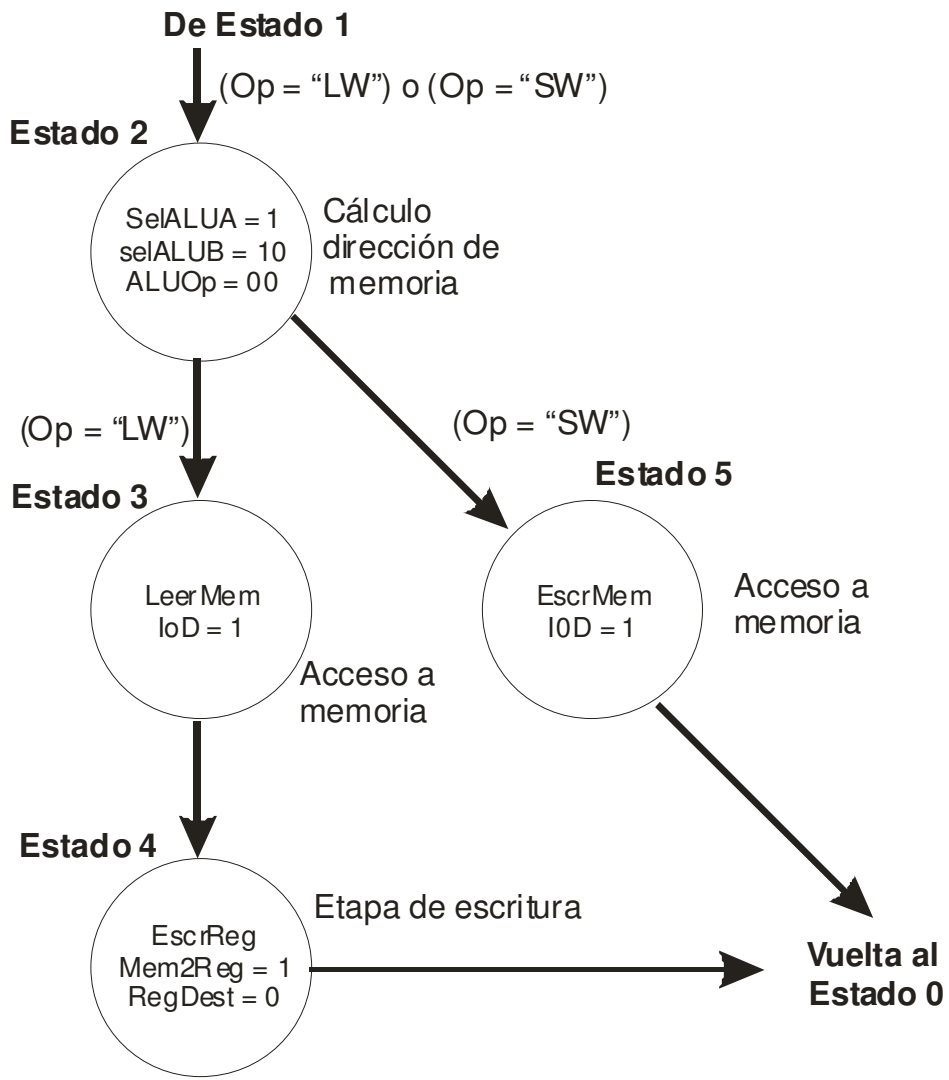
Estado 0. Cargar Instrucción



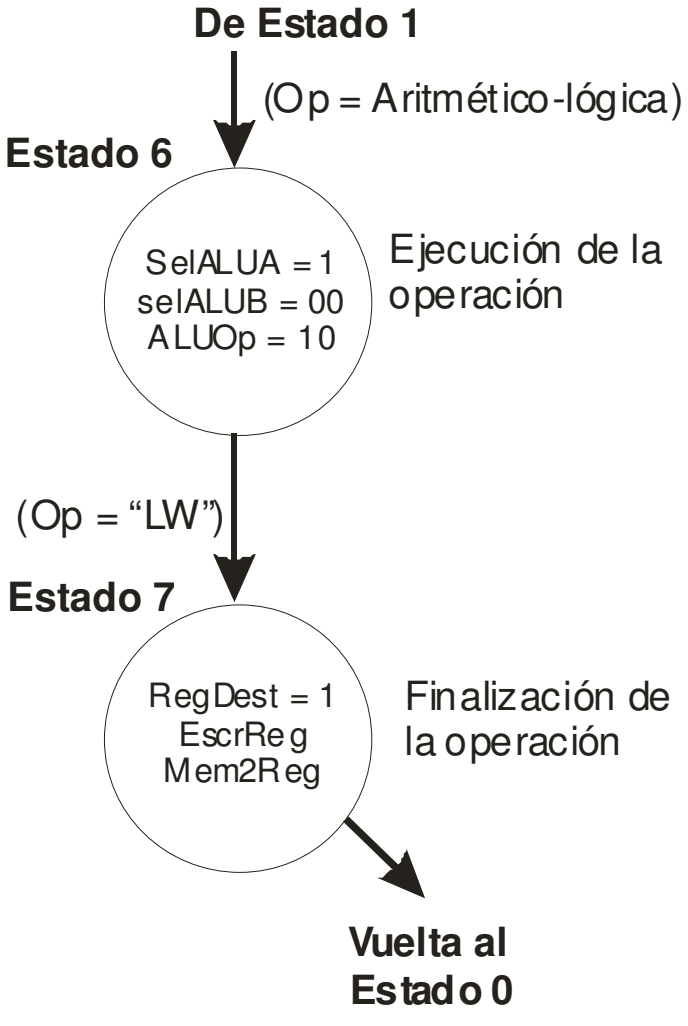
Estado 1. Decodificación



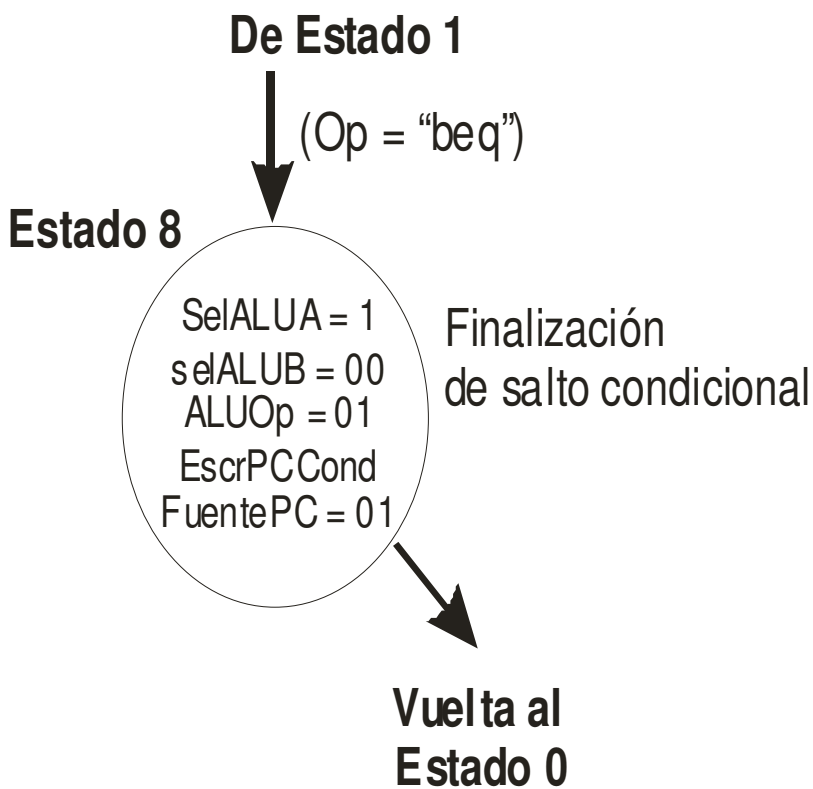
Control del acceso a memoria



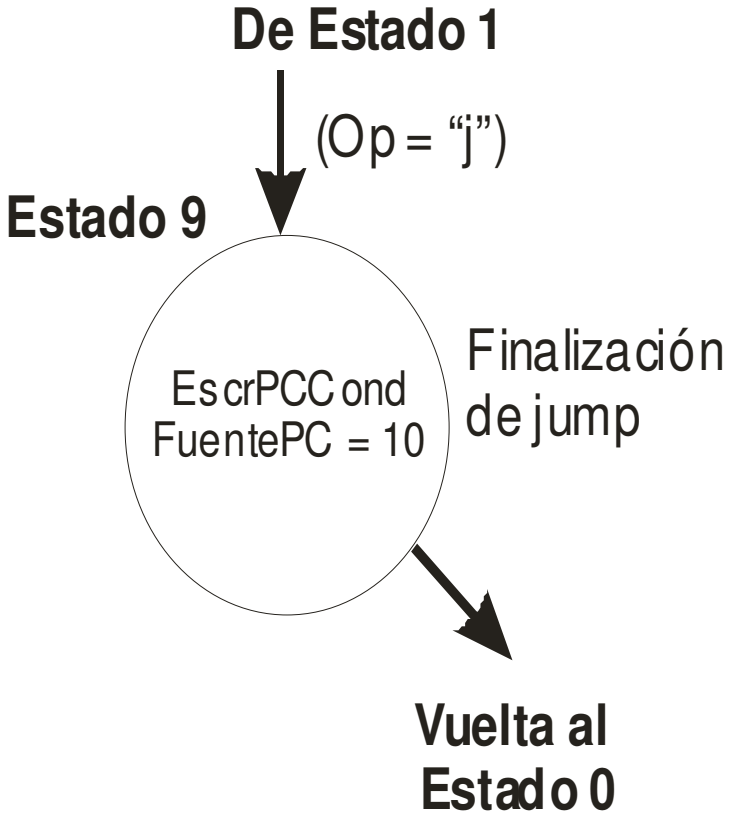
Control de operaciones aritmético-lógicas



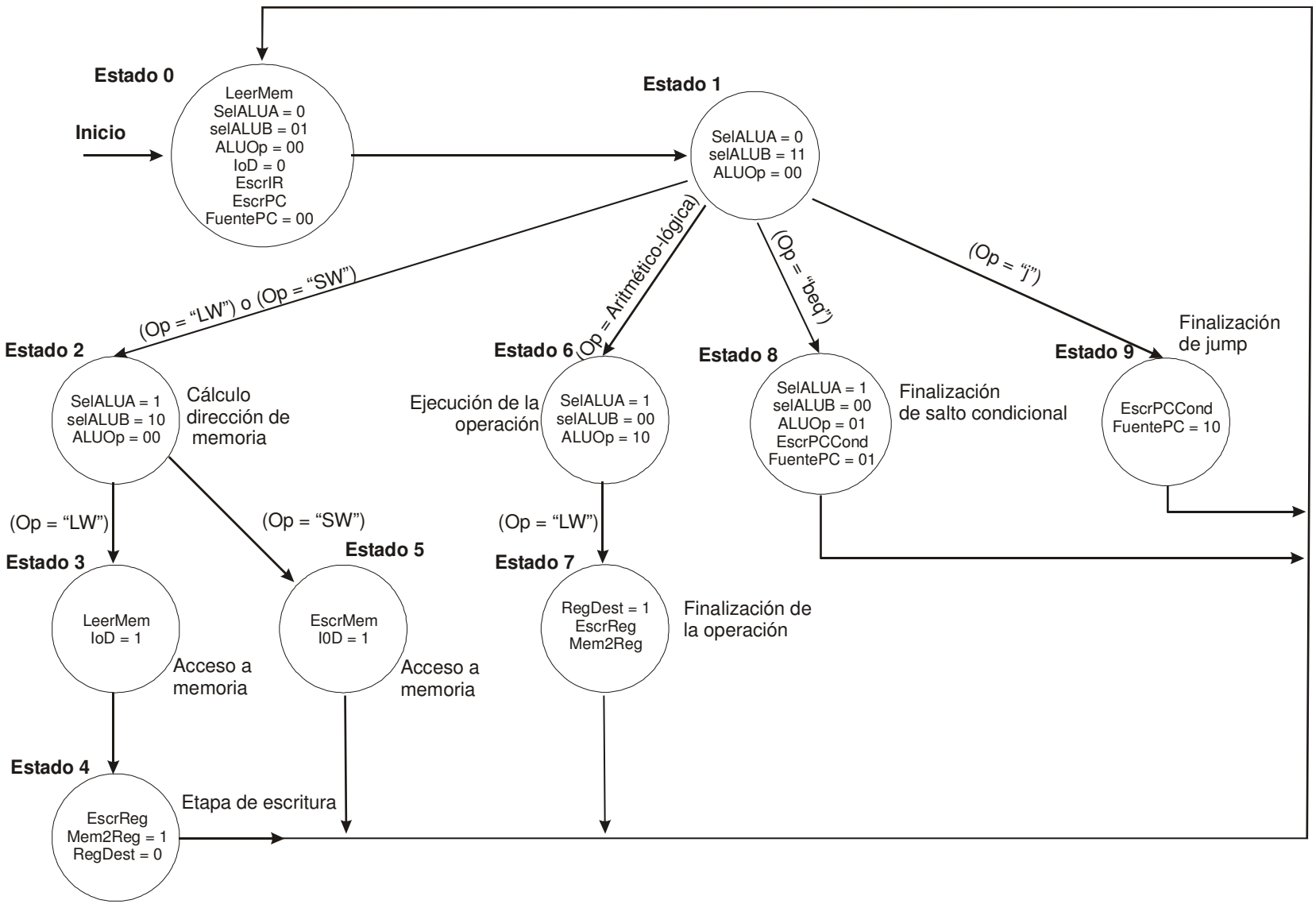
Control de beq



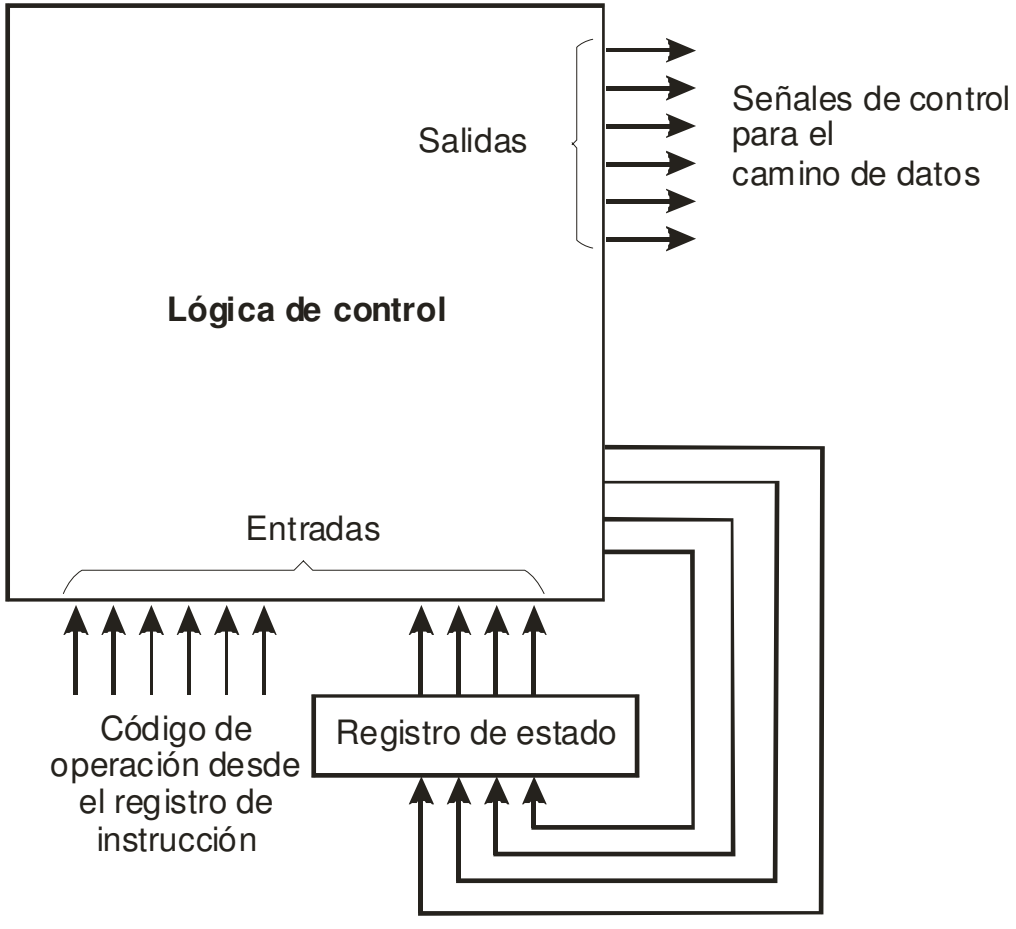
Control de jump



Máquina de estados completa



Implementación física

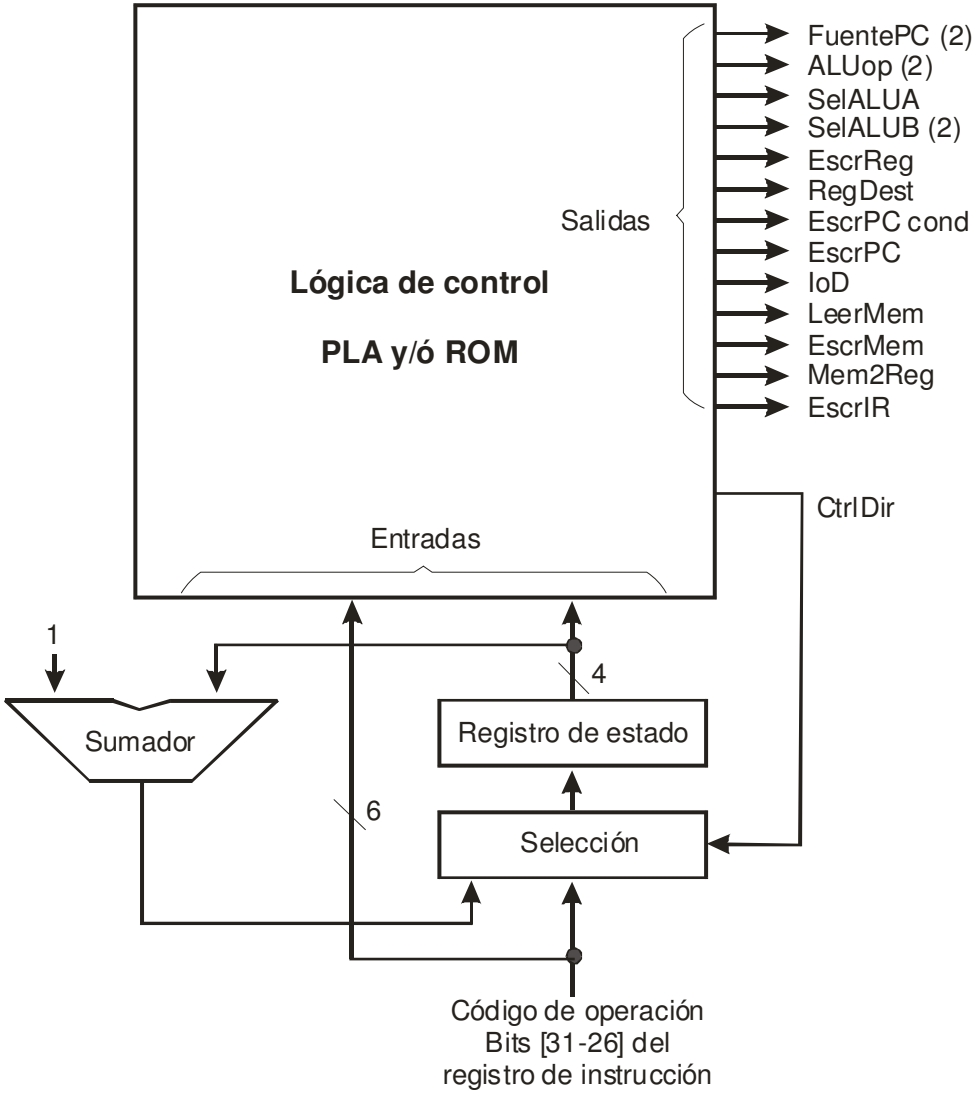


Concepto	Bits
Señales de control	16
Código de operación	6
Estado	4

Control microprogramado

- Cada grupo de señales causa la ejecución de una operación básica específica: microoperación.
- La interpretación y ejecución de una instrucción da lugar a una secuencia de operaciones máquina básicas (microoperaciones), cada una controlada por un grupo de señales de control, microinstrucción.
- Una secuencia de microinstrucciones constituye un microprograma.
- El código de operación de una instrucción máquina, cuando es decodificado, señala la microrrutina apropiada incluida en la memoria microprogramada.
- Las microinstrucciones suelen estar ubicadas en una ROM o en una PLA, por lo que pueden asignarse direcciones a las microinstrucciones.

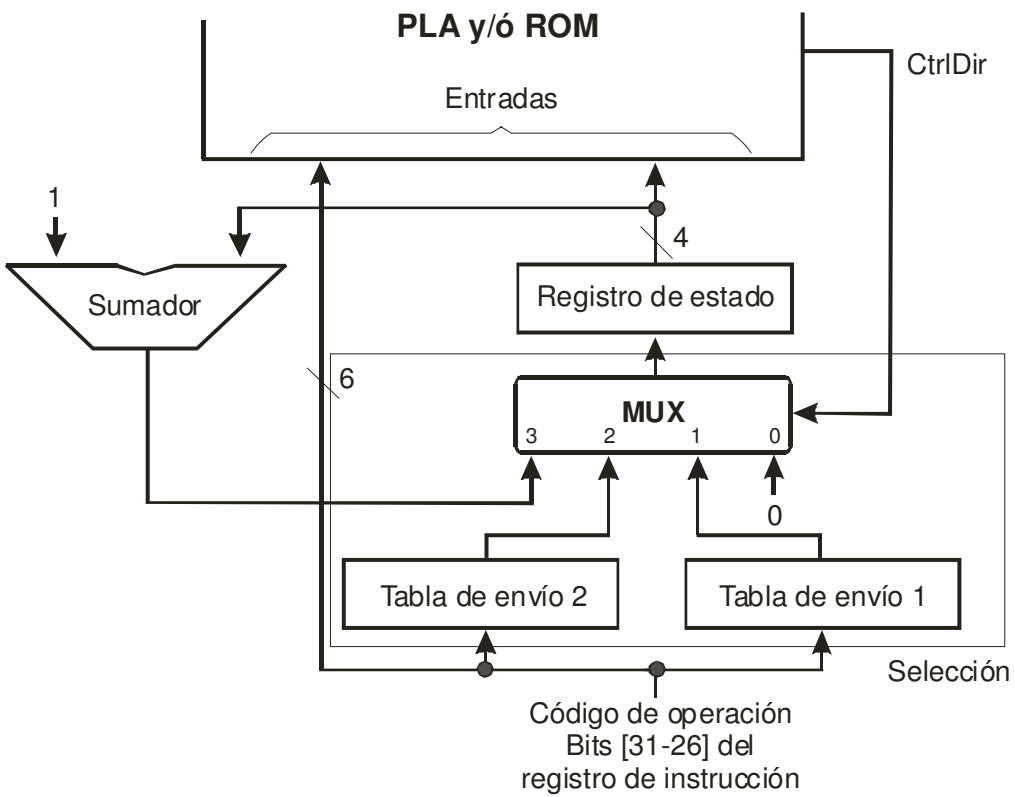
Control microprogramado



- El sumador avanza secuencialmente por los estados
- No en todos los casos se pasa al estado siguiente:
  - CtrlDir
  - Selección

Secuencia de estados

- √ Tras terminar una instrucción se ha de retornar al estado 0
- √ Las instrucciones sólo tienen algunos estados en común, después divergen
- √ Esta divergencia puede ocurrir en varios lugares en el diagrama de estados
- √ Necesitamos contemplar estos casos



- CtrlDir = 0
  - Nueva instrucción
- CtrlDir = 1
  - Tipo de instrucción
- CtrlDir = 2
  - Leer/escribir memoria
- CtrlDir = 3
  - Secuencia normal

Secuencia de estados

- Cada tabla de envío está asociada a un estado del que no tiene un único estado destino
- En el caso general crearemos una tabla (ROM/PLA) para cada estado con múltiples estados-destino
- En general, en un procesador complejo, los estados se seleccionarán de forma secuencial con pocas excepciones

Procesamiento de excepciones

- Una excepción es un suceso inesperado que se produce en el procesador, por ejemplo el desbordamiento aritmético
- Una interrupción es un suceso que provoca un cambio inesperado, pero se produce externamente al procesador
- Ejemplos de implementación de excepciones:
  - √ Instrucción indefinida
  - √ Desbordamiento aritmético
- Métodos para comunicar la causa de una excepción
  - √ Registro de estado (denominado registro Causa o *Cause Register*), que contiene un campo que indica la razón de la excepción. Es el método utilizado en la arquitectura MIPS. Se utiliza un único punto de entrada al sistema operativo para toda las excepciones
  - √ Interrupciones vectorizadas. La dirección a la que se transfiere el control viene determinada por la causa de la excepción. Por ejemplo,

Tipo de excepción	Dirección del vector de excepciones
Instrucción indefenida	0xC000 0000
Desbordamiento arimético	0xC000 0020

Procesamiento de excepciones

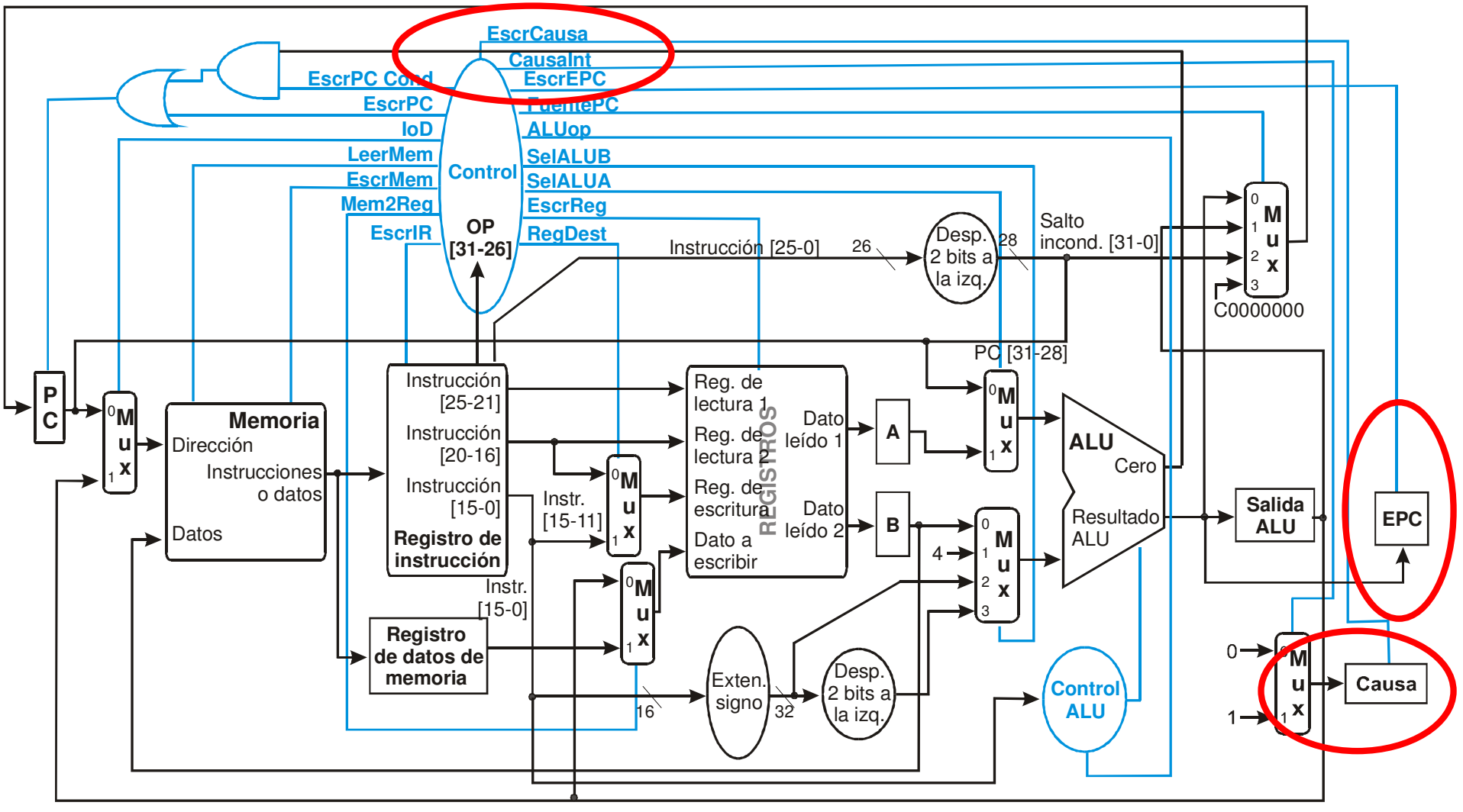
- Acciones a realizar:
  - √ Guardar la dirección de la instrucción causante en el registro contador de programa de la excepción (EPC)
  - √ Transferir el control al sistema operativo en alguna dirección específica
  - √ El sistema operativo ejecuta una rutina específica
  - √ Finalizar el programa o continuar con su ejecución, usando EPC para saber dónde retomar la ejecución

Procesamiento de excepciones

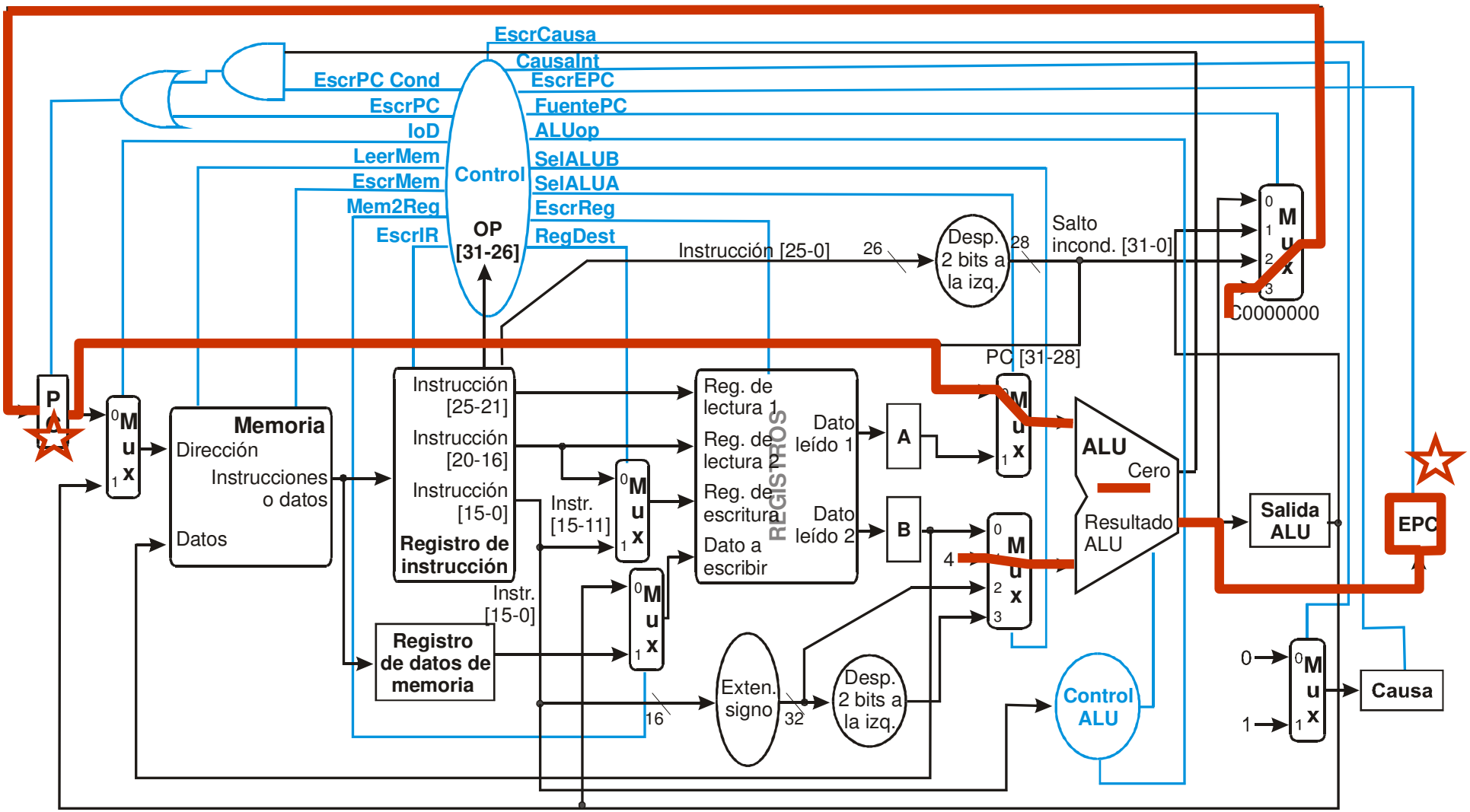
- Ejemplo de implementación:
  - √ Saltaremos a la dirección 0xC0000000
  - √ Necesitamos lo siguientes registros
    - Un registro de 32 bits para el EPC
    - Un registro de 1 bit para el Registro de Causa
  - √ Y las señales de control
    - Escribir en EPC
    - Escribir en Registro de Causa
    - Tipo de excepción (1 bit, pues solo consideramos dos tipos de excepciones)

Procesamiento de excepciones

- Camino de datos con los elementos necesarios para el procesamiento de excepciones:



Procesamiento de excepciones



Procesamiento de excepciones