Arquitectura de Computadoras para Ingeniería

(Cód. 7526)
1° Cuatrimestre 2018

Dra. Dana K. Urribarri
DCIC - UNS

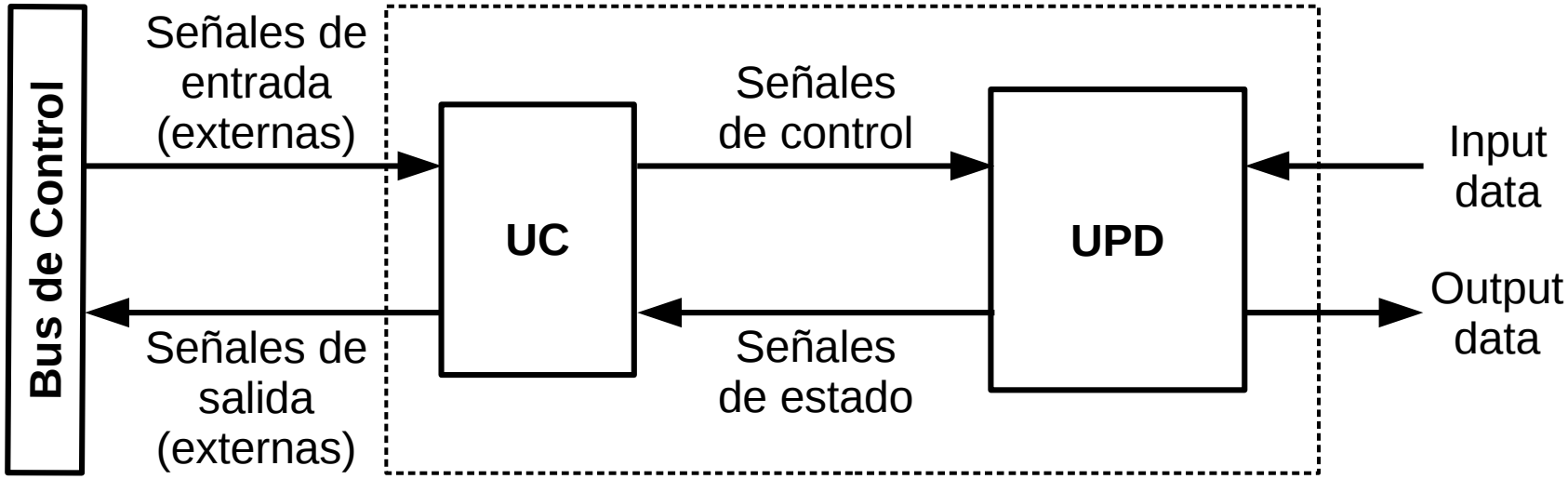
Control

Unidad de Control

- Provee las señales de control para la operación y coordinación de las componentes del procesador.

- El principal propósito de la Unidad de Control es decodificar las instrucciones y generar las señales que permitan realizar la operación deseada.

Interacción entre la Unidad de Control y la Unidad de Procesamiento de Datos



- La Unidad de Control (UC) indica la Unidad de Procesamiento de Datos (UPD) una secuencia de comandos.
- La UPD manipula los datos de acuerdo a lo indicado por la UC.

Tareas de la UC

1) Traer la instrucción de memoria (FETCH)
2) Decodificar de la instrucción (DECODE)
3) Ejecutar la instrucción (EXECUTE)

   Generar todas señales de control necesarias para indicarle a la UPD qué procesamiento aritmético-lógico debe realizar.

Funciones de la UC

La UC cumple dos funciones importantes:

- Secuenciamiento de las instrucciones:

  ¿Cómo se transfiere el control del procesador de una instrucción a otra?

- Interpretación de la instrucción:

  ¿Cómo se activan las señales de control que hacen que la UPD ejecute la instrucción?

Secuenciamiento de instrucciones

Secuenciamiento lineal

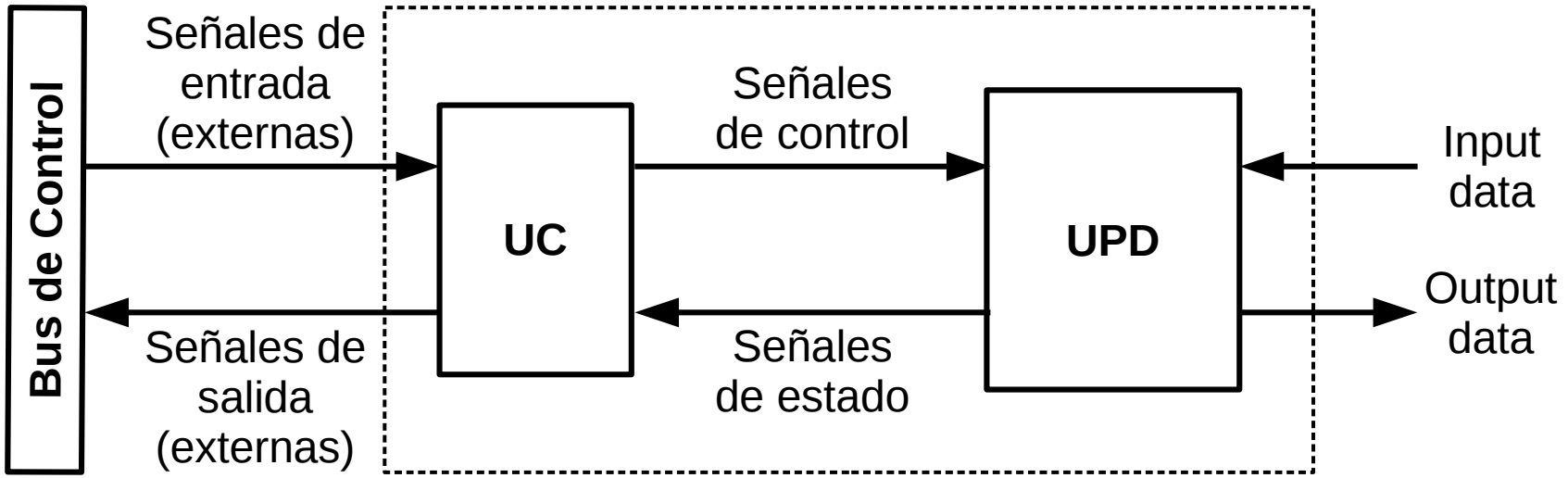
- Program counter

Secuenciamiento no lineal

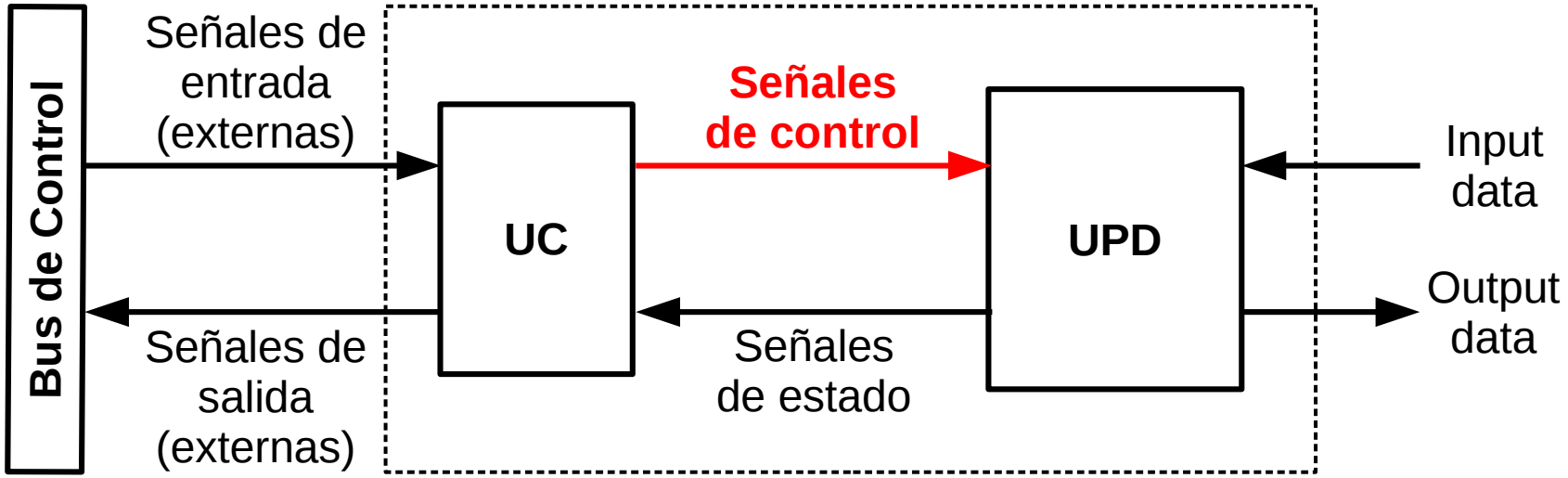
- Branch – jump
- Transferencia de control entre (sub)programas
  - Llamada a subrutina: Transferencia temporaria del control de *A* a *B*, iniciado por *A*
  - Interrupción: Transferencia temporaria del control de *A* a *B*, iniciado por *B* o un dispositivo asociado a *B*
  - Almacenar
    - La dirección de retorno
    - Posibles parámetros de *A* a *B*
    - Variables locales a *B*

Interpretación de la instrucción

- La UC interpreta la instrucción para determinar qué señales de control debe transmitir.
- Estas señales de control se transmiten a través de *líneas de control.*
- Se distinguen 4 grupos de señales de control:

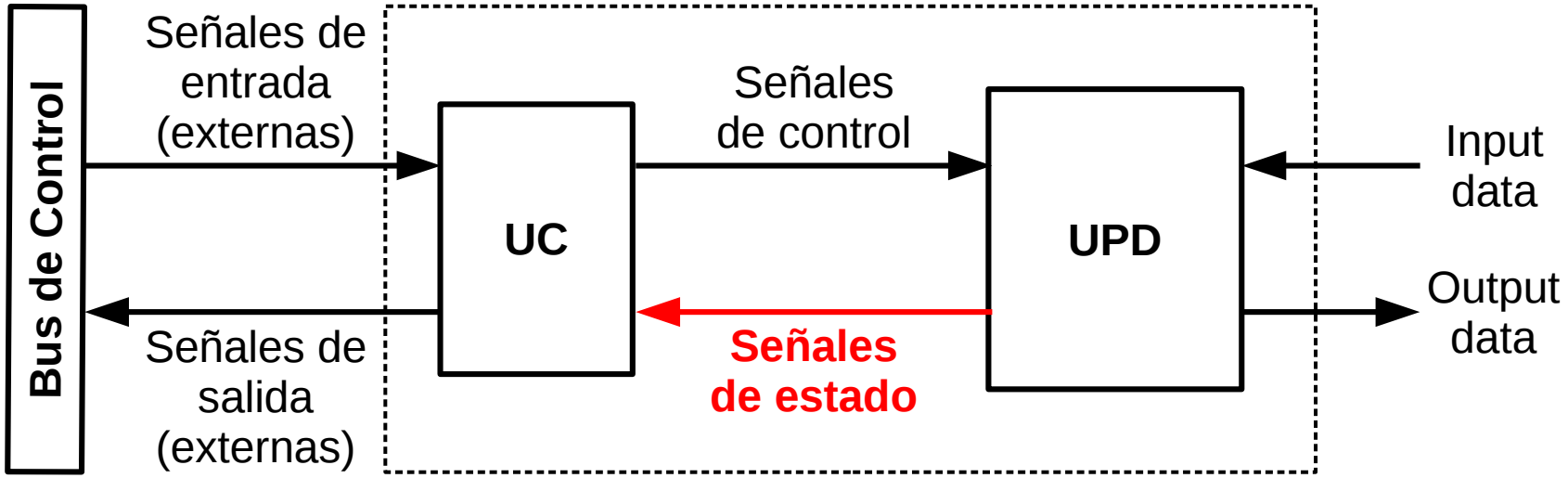


Interpretación de la instrucción



- Señales de control
  - Objetivo principal de la UC.
  - Señales que controlan la operación de la UPD

Interpretación de la instrucción

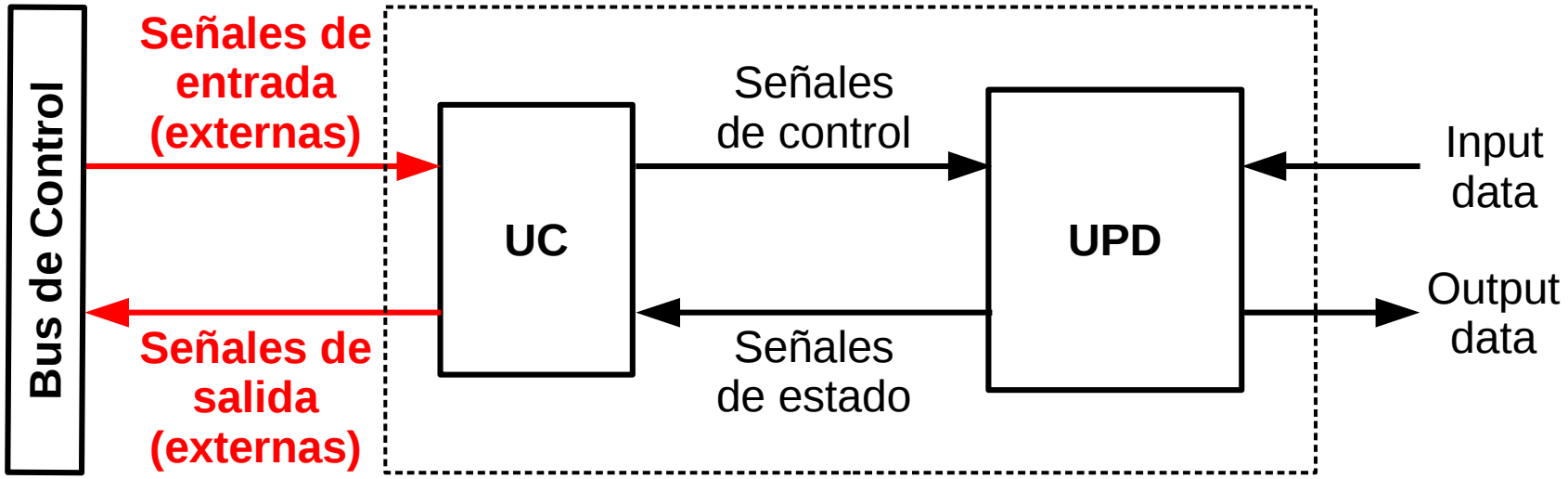


- Señales de estado

  Permiten que la UC tome decisiones basadas en los datos.

  Le permiten a la UPD indicar errores en el procesamiento de los datos (x ej. overflow).

Interpretación de la instrucción



Las señales externas se usan para sincronizar con otros controladores.

- Señales de entrada (externas)

  Señales que recibe la UC a través del bus de control (señales de *start*, *stop* y temporizado).

- Señales de salida (externas)

  Incluye señales de control a la memoria y señales de control a los módulos de I/O

Implementación de la Unidad de Control

Implementación de la UC

La UC tiene típicamente dos partes:

- Una parte combinacional que carece de estado (por ej. para las decodificaciones, diseño único-ciclo)
- Otra parte secuencial para el secuenciamiento de las instrucciones y para el control principal de un diseño múltiple-ciclo.
  - ROM
  - PLA
  - Contador
  - Microprogramación

Implementación de la UC

La UC tiene típicamente dos partes:

- Una parte combinacional que carece de estado (por ej. para las decodificaciones, diseño único-ciclo)
- Otra parte secuencial para el secuenciamiento de las instrucciones y para el control principal de un diseño múltiple-ciclo.
  - ROM
  - PLA
  - Contador
  - Microprogramación

Control Cableado

Implementación de la UC

La UC tiene típicamente dos partes:

- Una parte combinacional que carece de estado (por ej. para las decodificaciones, diseño único-ciclo)
- Otra parte secuencial para el secuenciamiento de las instrucciones y para el control principal de un diseño múltiple-ciclo.
  - ROM
  - PLA
  - Contador
  - Microprogramación

Control Microprogramado

Control cableado

Ejemplo

- Tenemos una ALU de la cual nos interesan las siguientes operaciones:

Control de la ALU	Función
0000	AND
0001	OR
0010	suma
0110	resta
0111	menor que

Ejemplo

- Las siguientes 4 instrucciones utilizan la ALU:

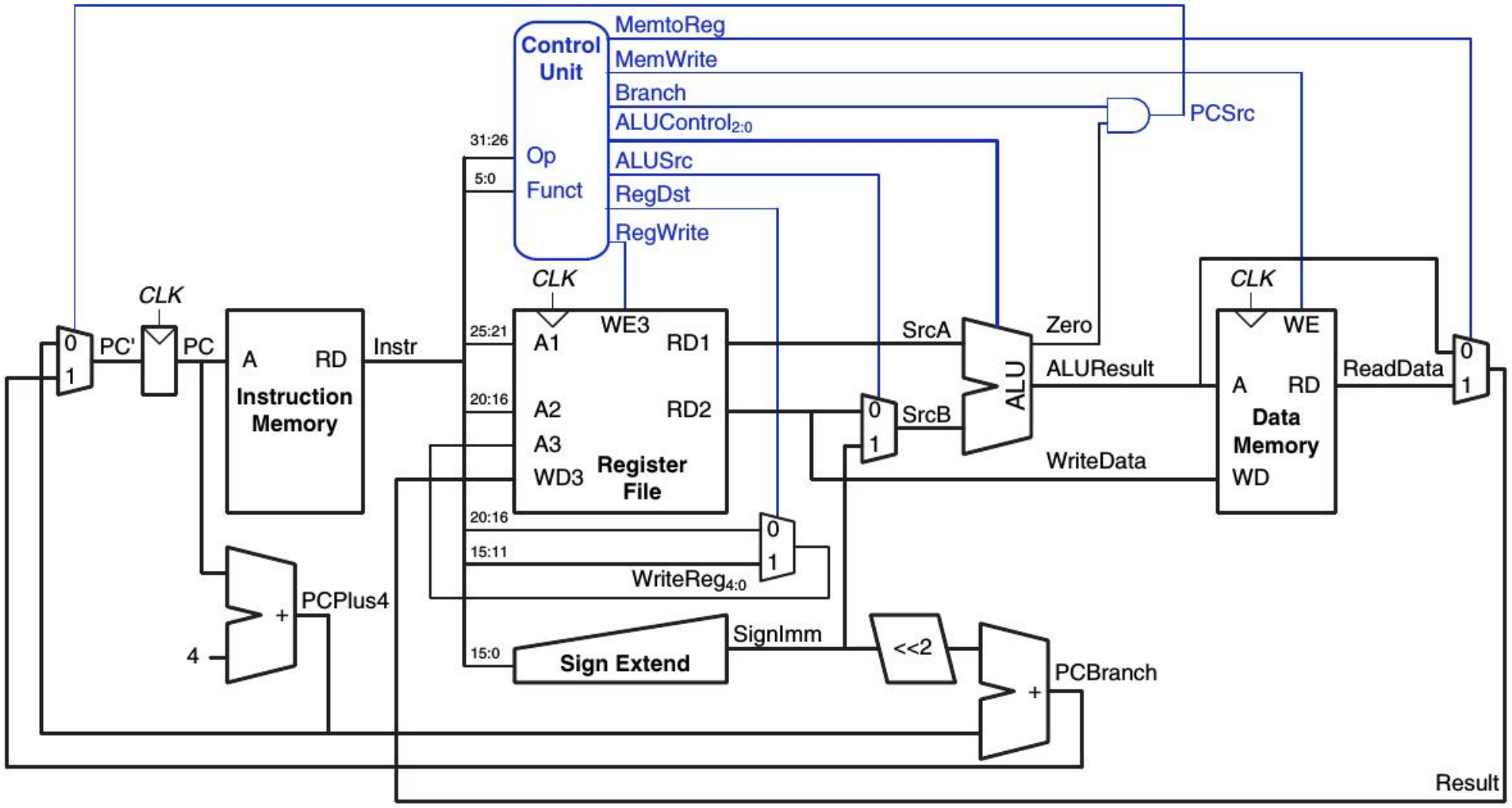
Instrucción	Operación
R-type	cualquiera
load	suma
store	suma
Branch on equal	resta

  - R-type: Operaciones registro a registro, dos fuentes y uno destino. Tienen el siguiente formato:

Opcode	R_S	R_T	R_D	...	Funct
6 bits	5 bits	5 bits	5 bits	...	6 bits

UC Combinacional

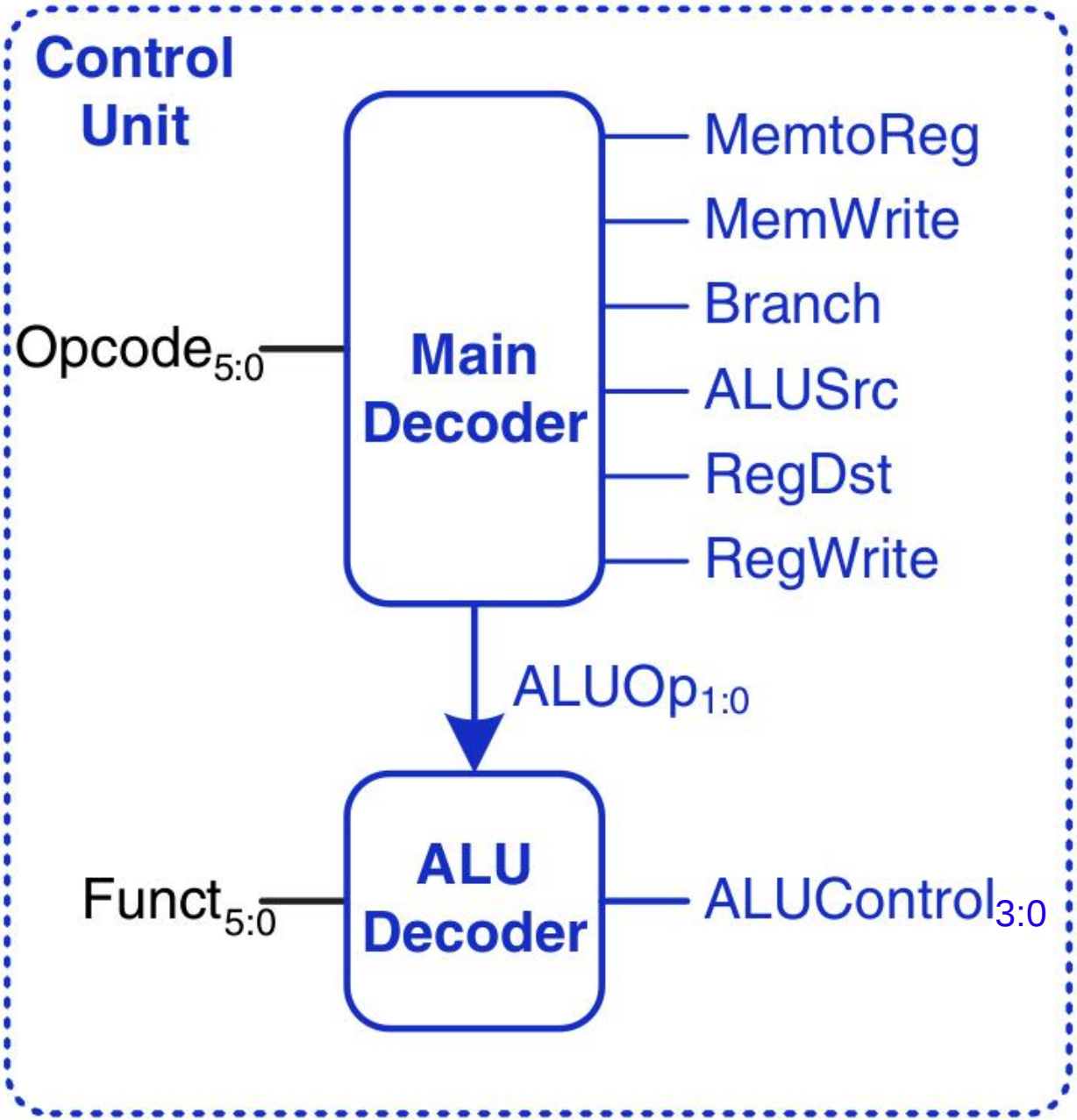
- Procesador único-ciclo



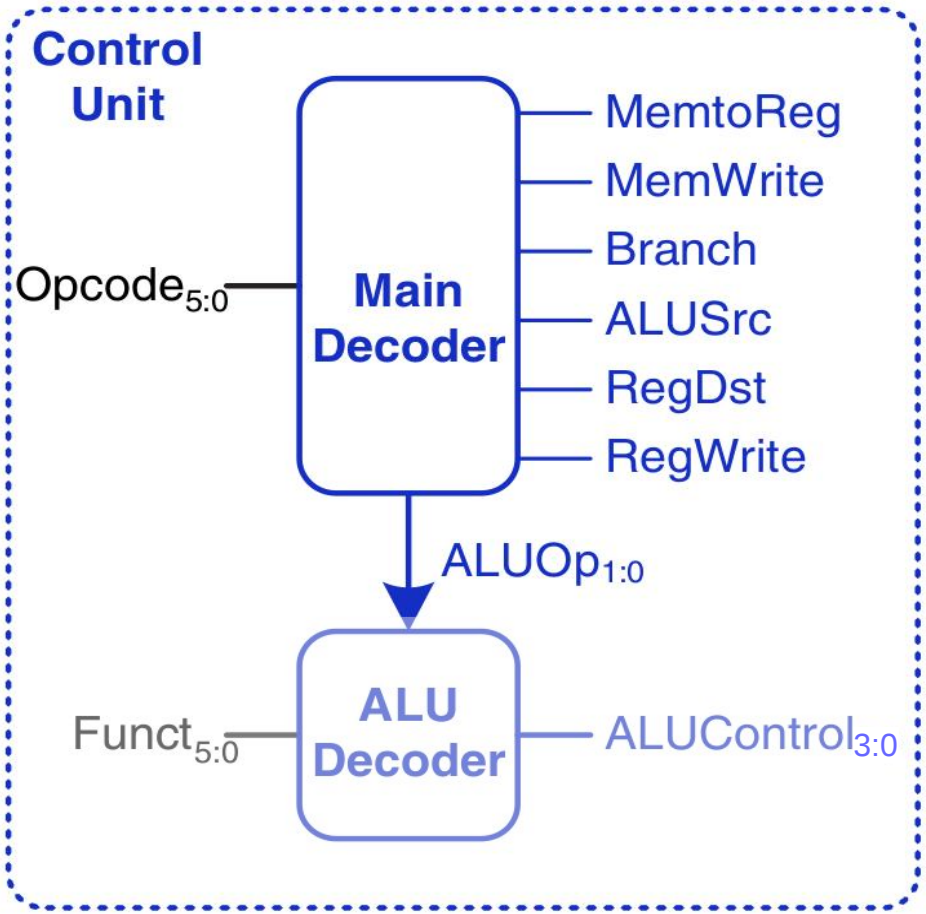
UC Combinacional

- La UC computa las señales de control basándose en el *opcode* y en el campo *funct* de la instrucción.
- La mayoría de la información de control proviene del *opcode*
- Diseño simplificado. Dos bloques.

Estructura interna de la UC combinacional para el Procesador único-ciclo



UC Combinacional



Main decoder

Salidas a partir del opcode.

Incluye una señal de 2 bits *ALUOp* para el siguiente bloque.

$ALUOp_{1:0}$	Significado
00	suma
01	resta
10	mirar campo *funct*
11	N/a

UC Combinacional

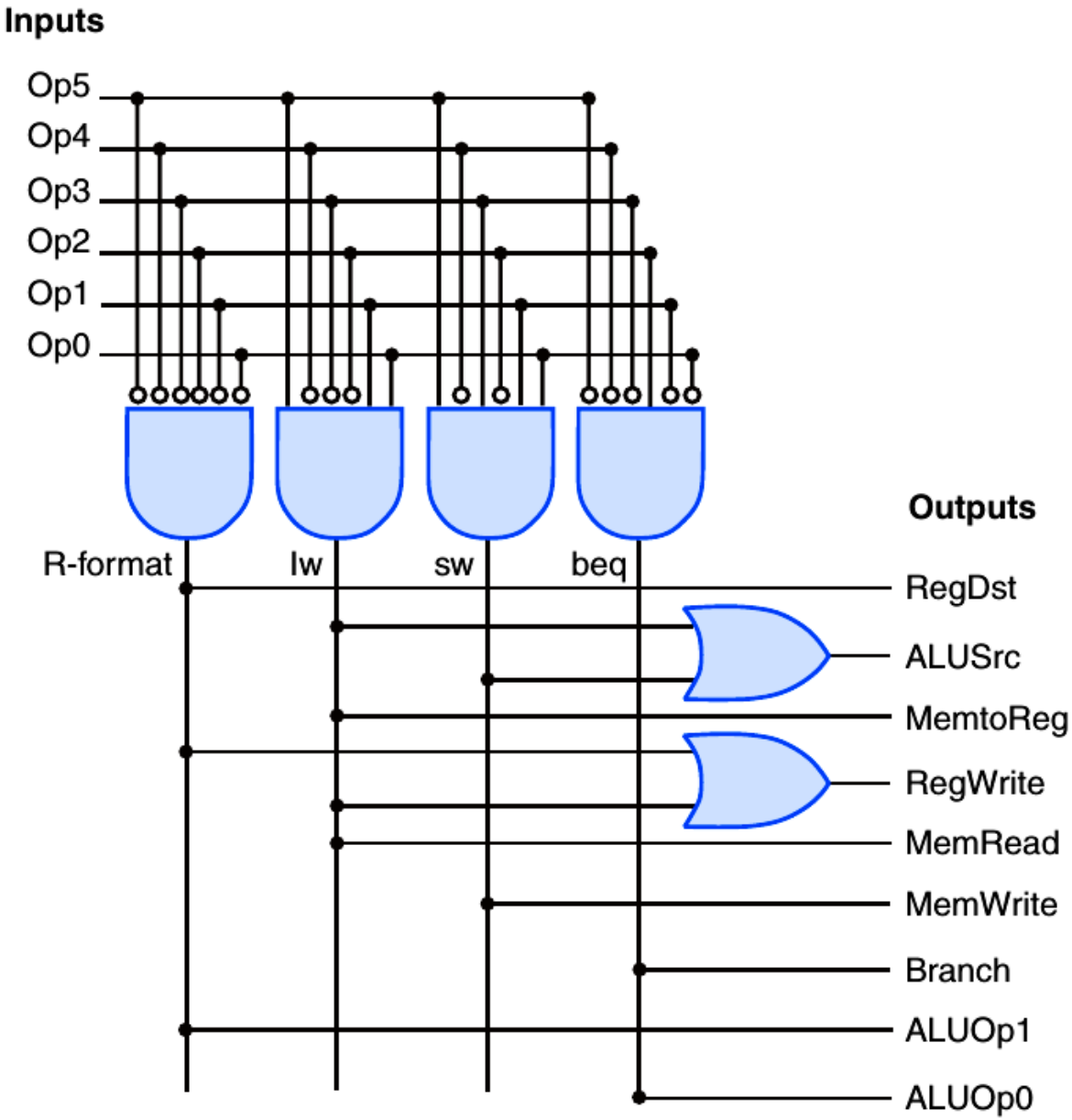
Main Decoder: Tabla de verdad:

- Entradas: 6 bits del opcode de cada instrucción
- Salidas:
  - 6 señales de control
  - 2 bits de ALUOp

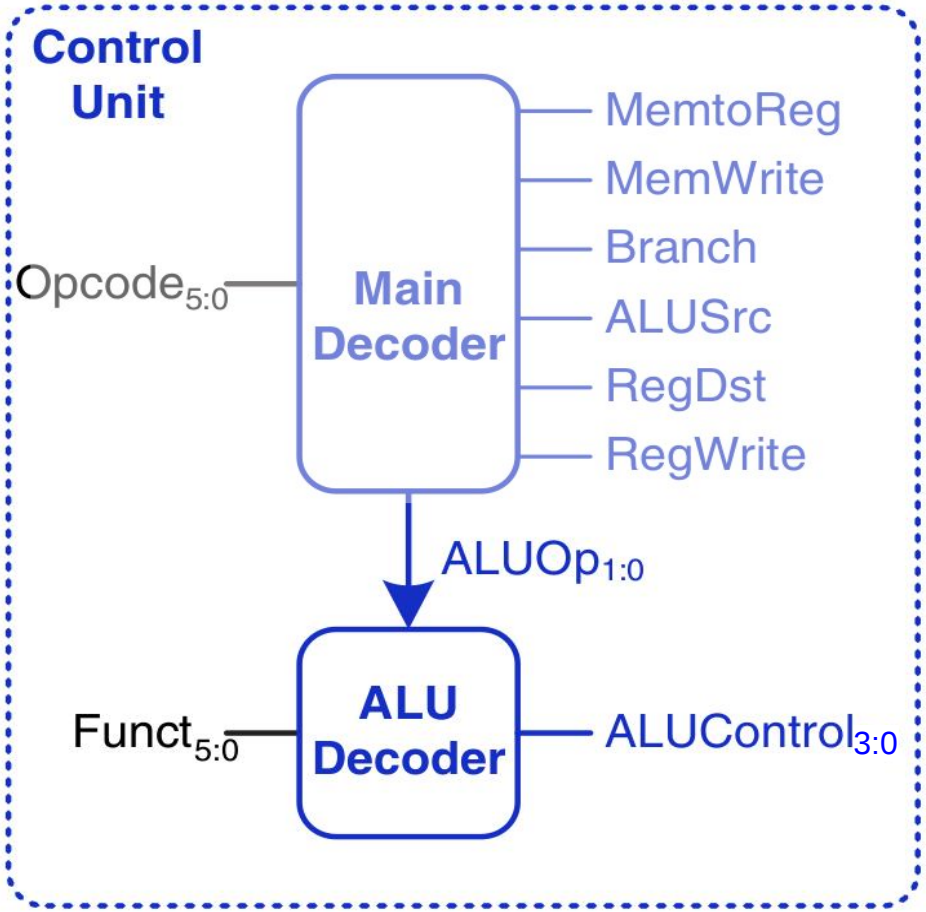
Instrucción	Opcode	RegWrite	RegDst	ALUSrc	Branch	MemWrite	MemToReg	ALUOp
R-type	000000	1	1	0	0	0	0	10
load	100011	1	0	1	0	0	1	00
store	101011	0	X	1	0	1	X	00
branch	000100	0	X	0	1	0	X	01

UC Combinacional

Main Decoder



UC Combinacional



ALU Decoder

A partir de la señal *ALUOp* y el campo *funct*, computa la operación de entrada a la ALU (*ALUControl*)

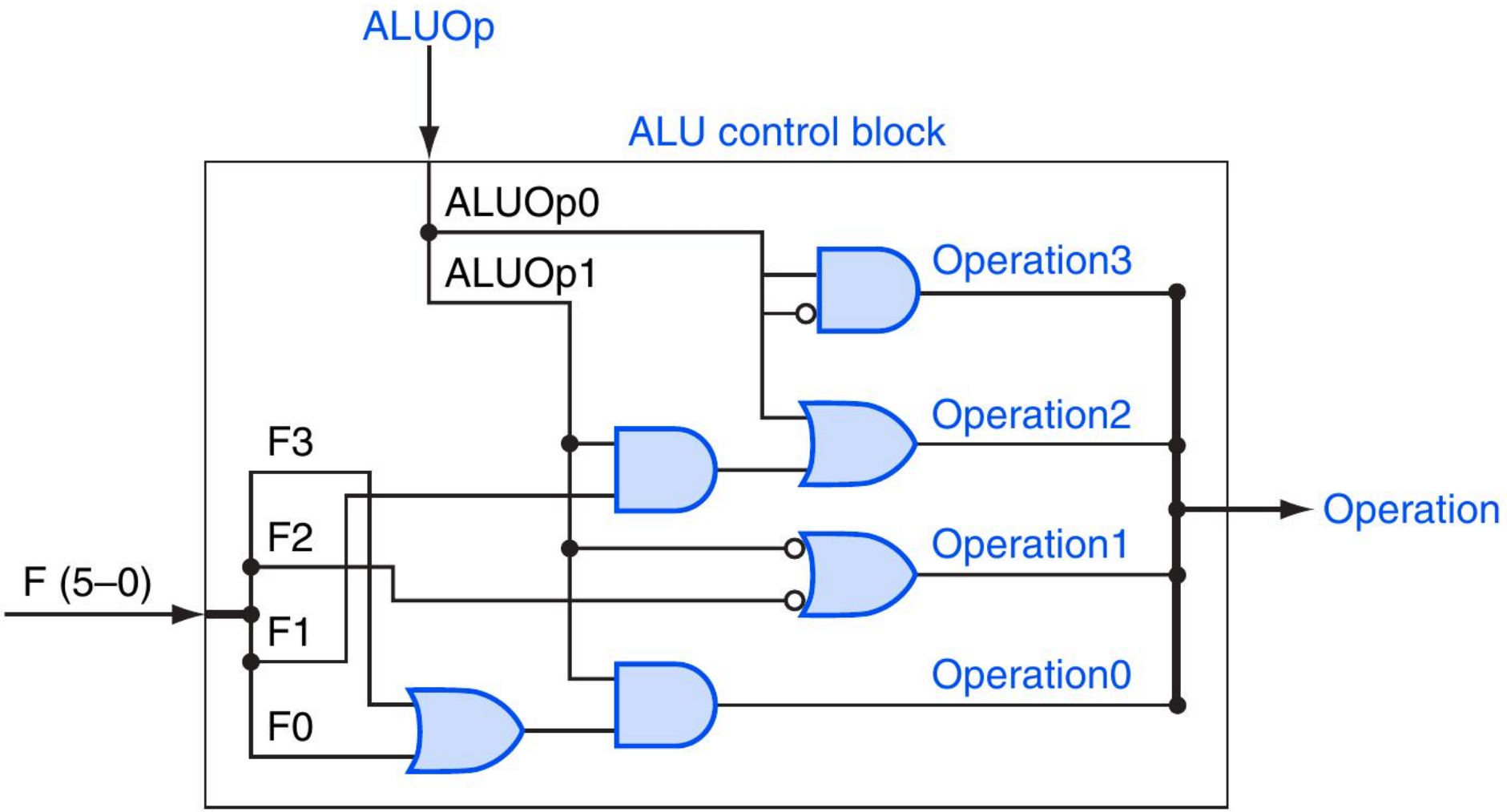
UC Combinacional

ALU Decoder: Tabla de verdad

ALUOp		Campo *Funct*						Operación
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
0	1	X	X	X	X	X	X	0110
1	0	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	0	X	X	0	1	0	0	0000
1	0	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

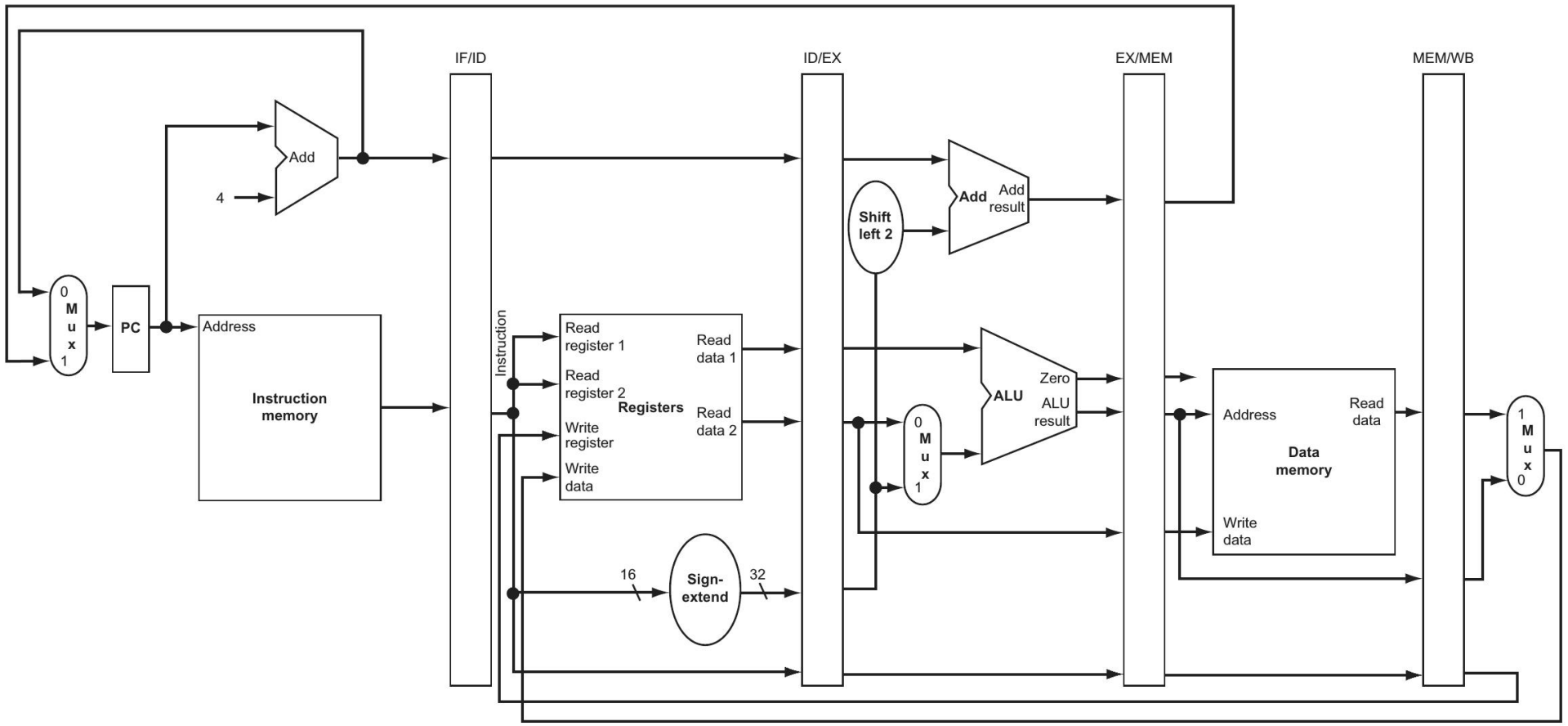
UC Combinacional

ALU Decoder



Control del Pipeline

Procesador en Pipeline



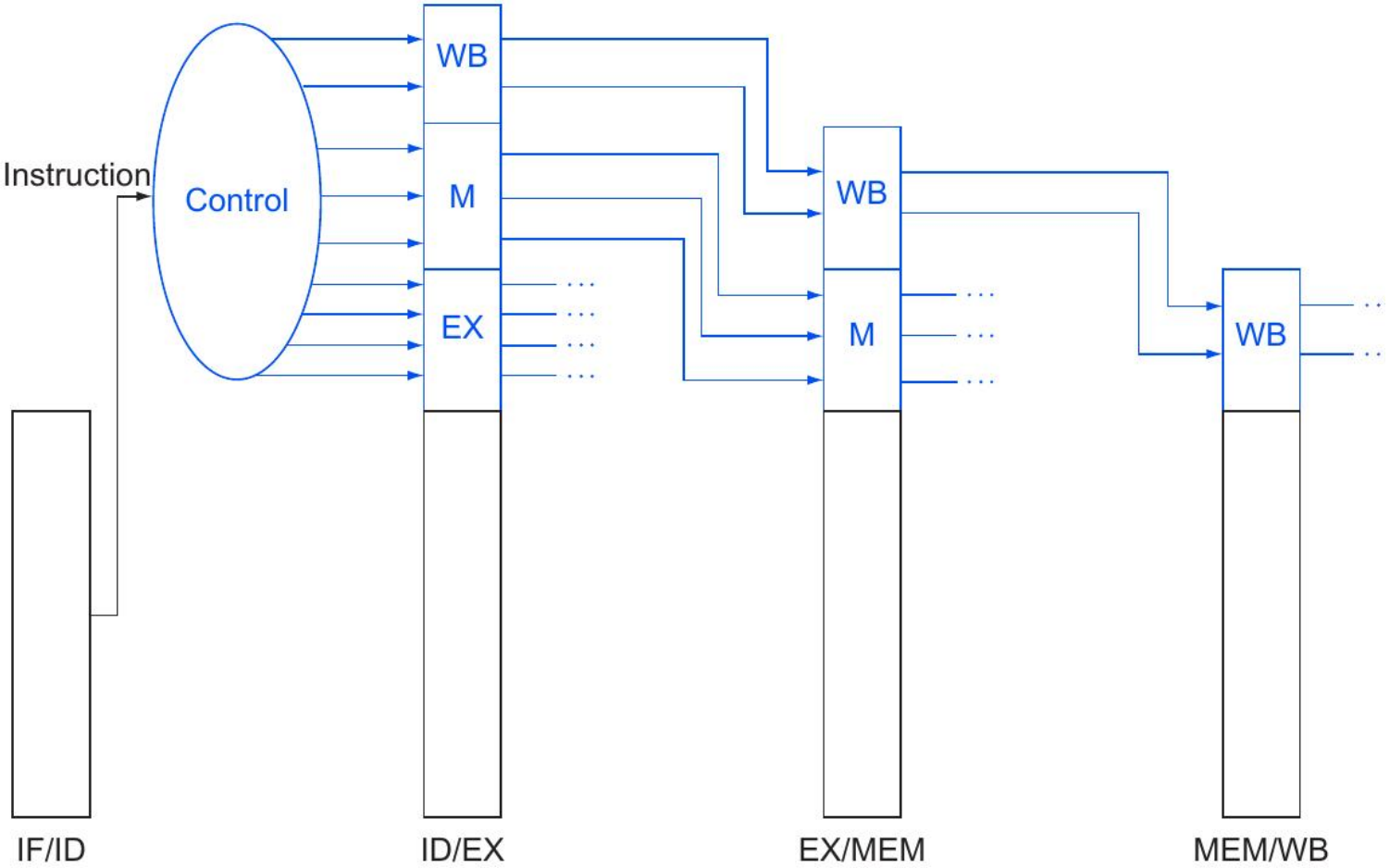
FETCH DECODE EXECUTE MEMORY WB

Control del Pipeline

- Tiene las mismas señales de control que el procesador único-ciclo.
- Utiliza la misma UC
- Las señales de control se activan en la etapa DECODE pero se utilizan en las etapas siguientes.

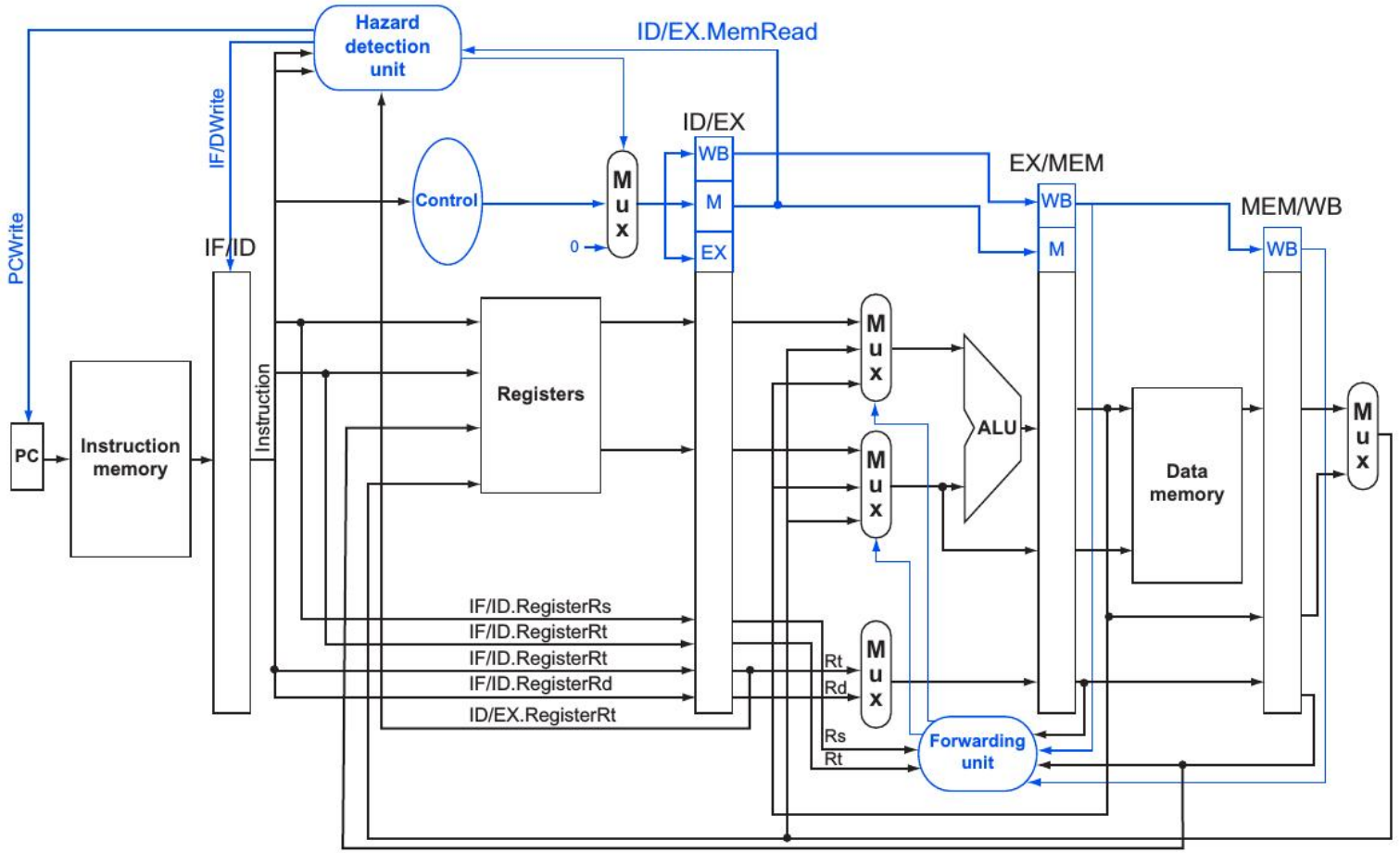
Control del Pipeline

- Se extienden los registros interetapas para que incluyan la información de control.



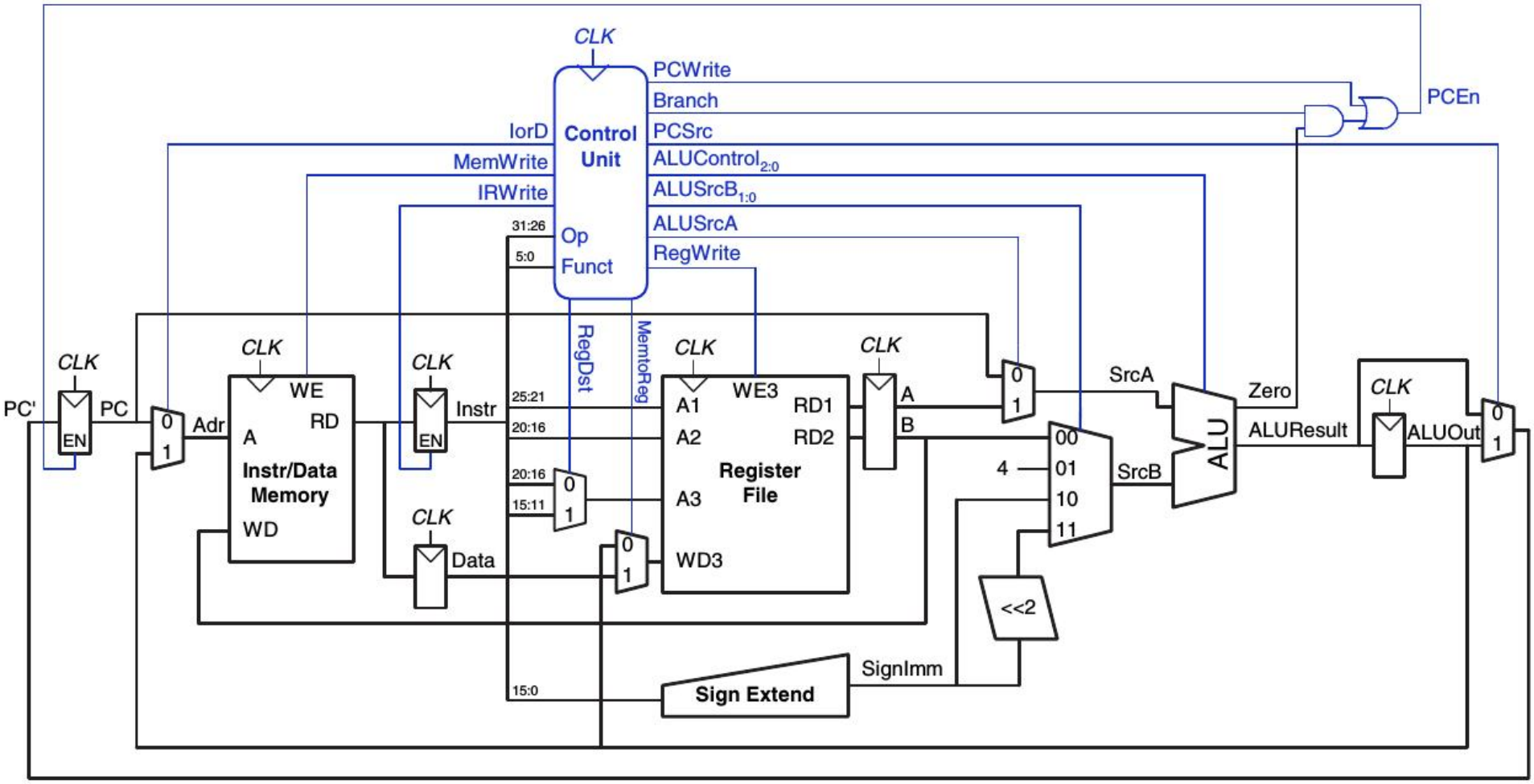
Control del Pipeline

- Además de registros interetapas, agrega multiplexores y lógica de control para resolver *hazards* y *forwarding*.

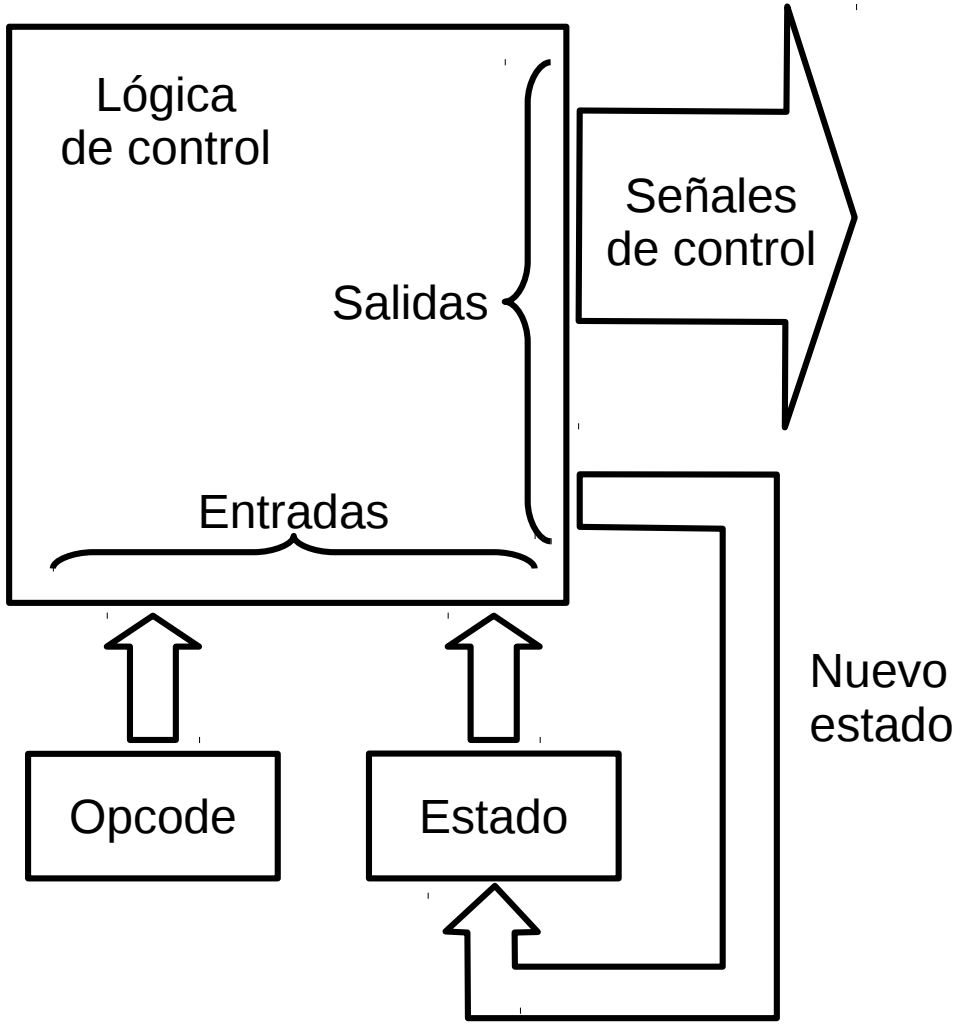


UC con una Máquina de estados

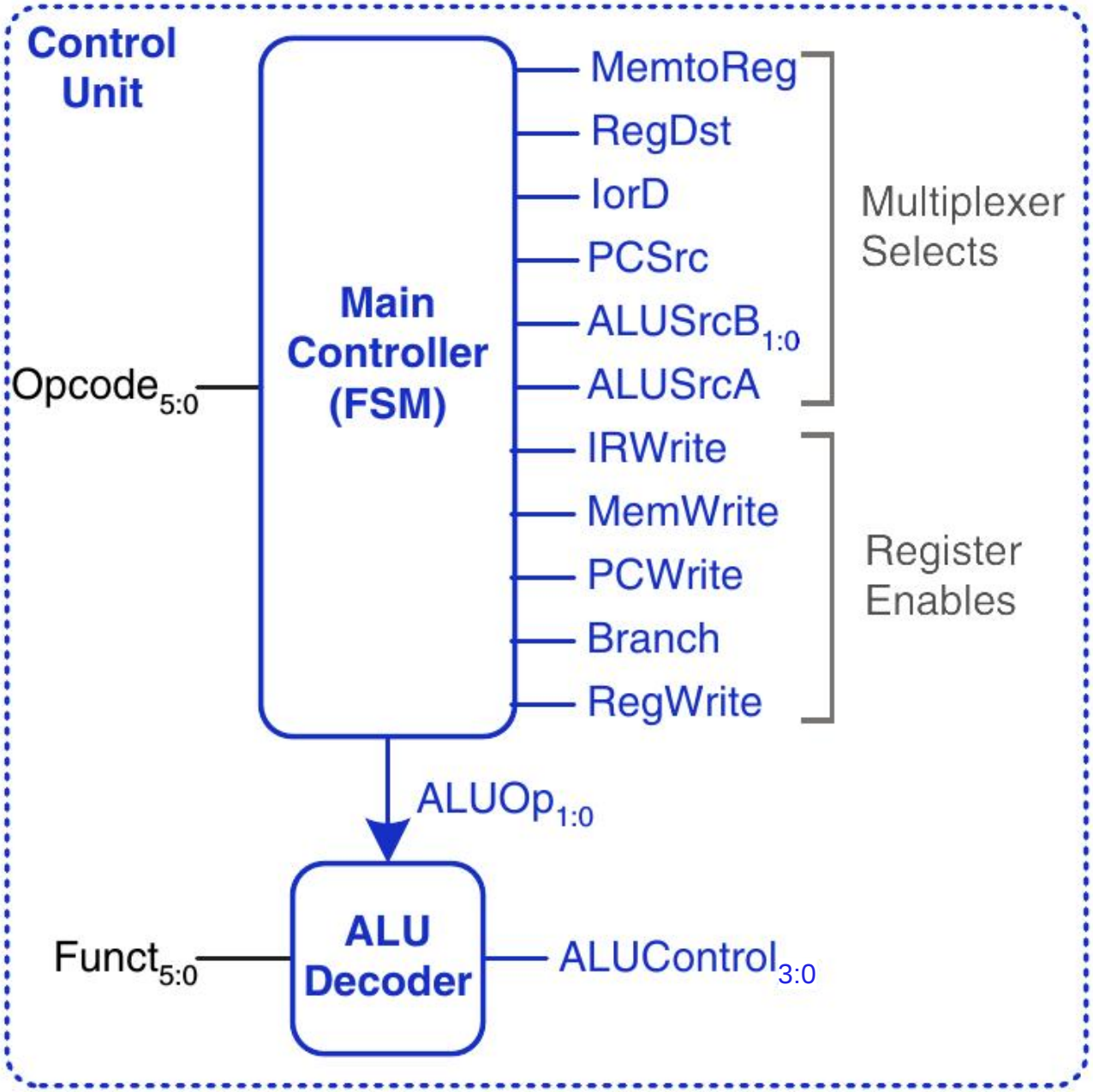
- Procesador múltiple-ciclo



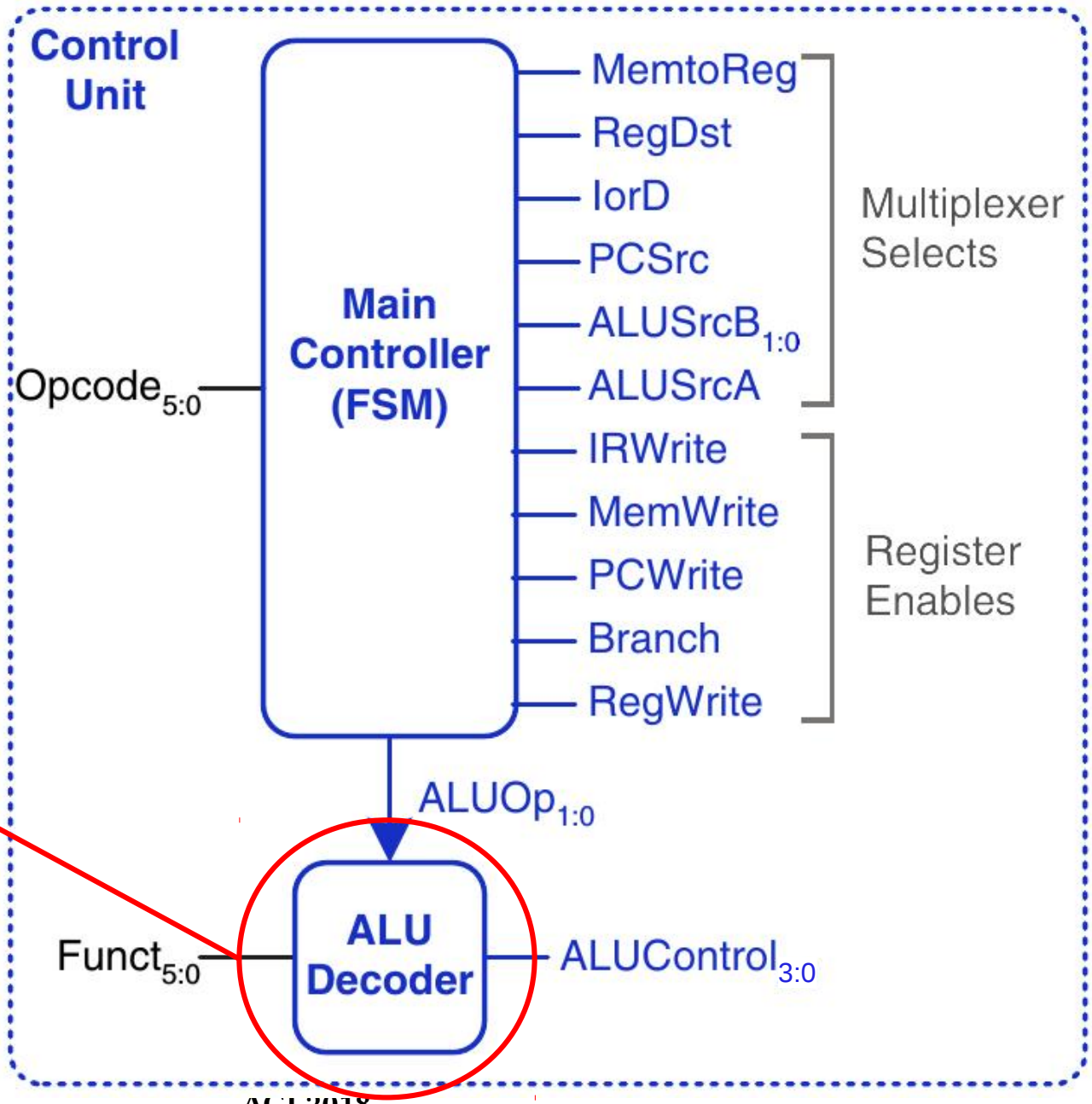
UC con una Máquina de estados



Estructura interna de la UC combinada para un Procesador múltiple-ciclo

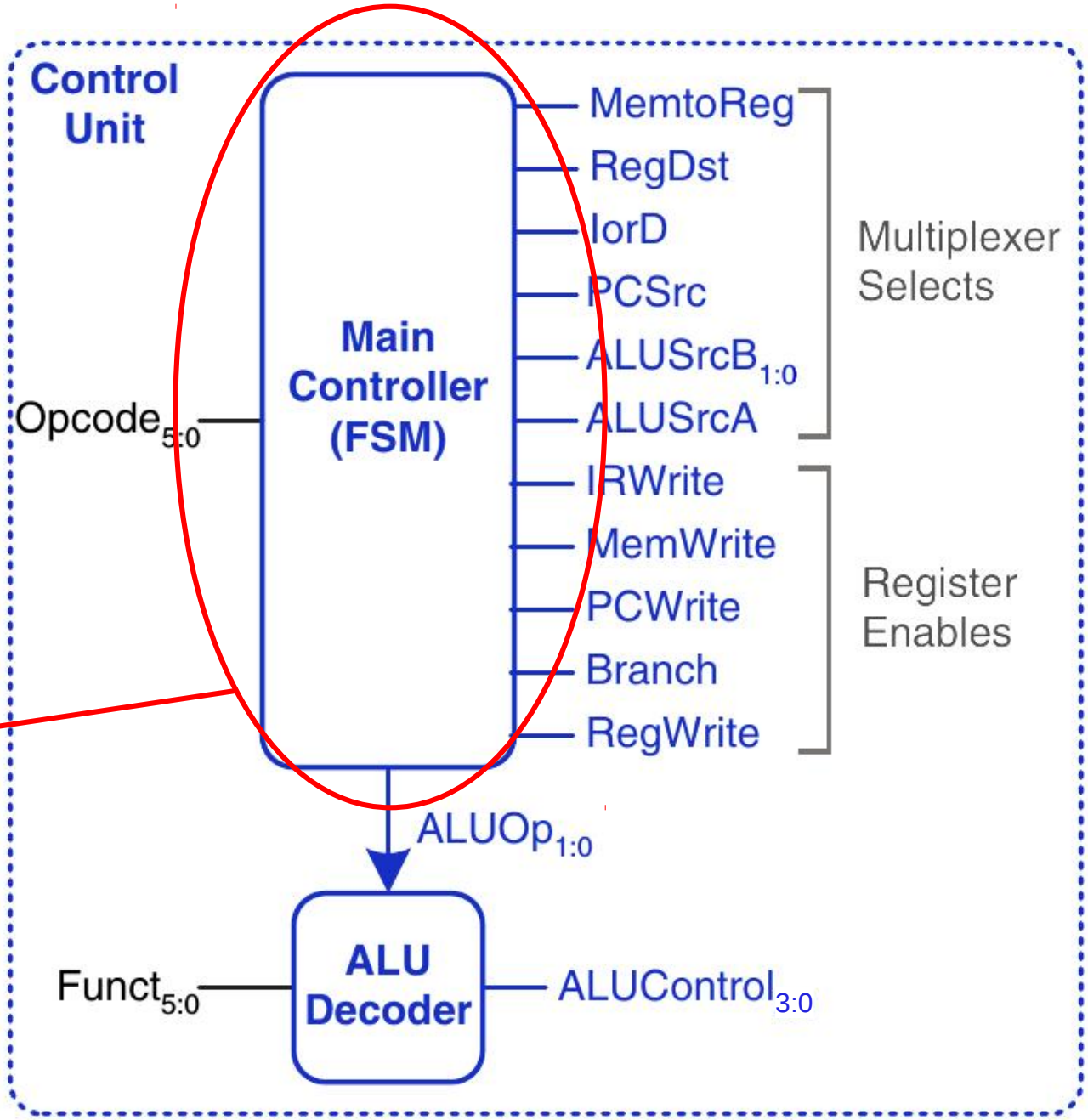


Estructura interna de la UC combinada para un Procesador múltiple-ciclo



Circuito combinacional

Estructura interna de la UC combinada para un Procesador múltiple-ciclo



Circuito secuencial

UC con una Máquina de estados

El comportamiento de la UC se puede representar con una tabla de estados:

Estado	I_1	I_2	...	I_m
S_1	$S_{1,1}$, $z_{1,1}$	$S_{1,2}$, $z_{1,2}$	...	$S_{1,m}$, $z_{1,m}$
S_2	$S_{2,1}$, $z_{2,1}$	$S_{2,2}$, $z_{2,2}$	...	$S_{2,m}$, $z_{2,m}$
...				
S_n	$S_{n,1}$, $z_{n,1}$	$S_{n,2}$, $z_{n,2}$	...	$S_{n,m}$, $z_{n,m}$

- S_i corresponde al estado interno de la máquina
- I_j corresponde al opcode de la instrucción j
- $S_{i,j}$ corresponde al estado que pasa la máquina estando en el estado S_i mientras ejecuta I_j
- $z_{i,j}$ corresponde a las señales de control cuando ejecutando I_j se está en el estado S_i.
- Para n estados se necesitan P bits, P = $\lceil \text{Log}_2\ n \rceil$
- Para m opcodes se necesitan Q bits, Q = $\lceil \text{Log}_2\ m \rceil$

UC con una Máquina de estados

Con una ROM

	r salidas de control				*P* bits para el prox. estado			
	z_1	z_2	...	z_r	$Estado_0$	$Estado_1$	...	$Estado_P$
$n \times m \approx 2^{P+Q}$ entradas	{0,1,X}	{0,1,X}	...	{0,1,X}	{0,1,X}	{0,1,X}	...	{0,1,X}
	{0,1,X}	{0,1,X}	...	{0,1,X}	{0,1,X}	{0,1,X}	...	{0,1,X}
	...	...	...	...				
	{0,1,X}	{0,1,X}	...	{0,1,X}	{0,1,X}	{0,1,X}	...	{0,1,X}

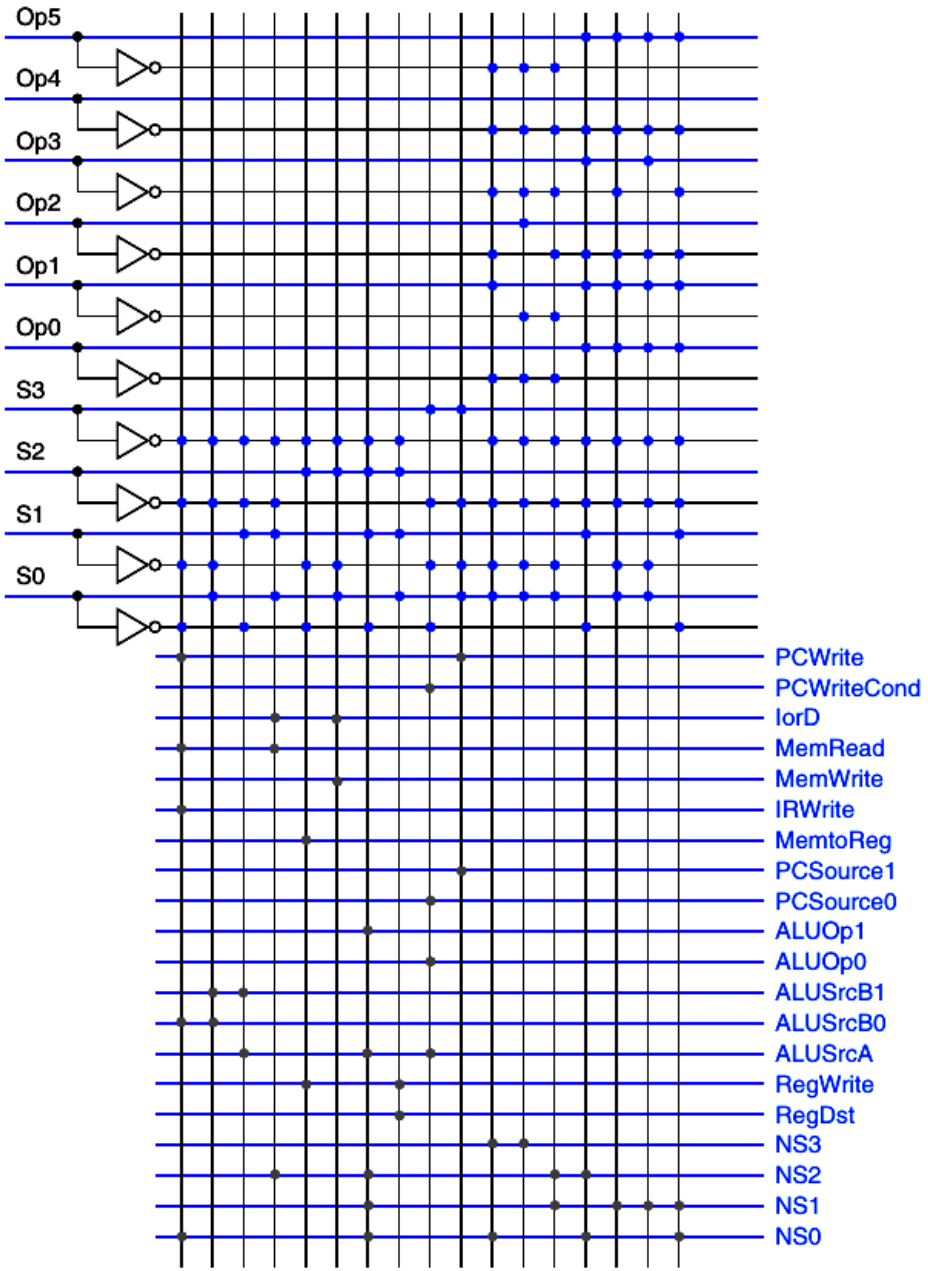
Si tenemos

- 20 bits de salidas (4 bits del siguiente estado, 2 bits de ALUOp y 16 bits de señales de control)
- 10 bits de entrada (6 bits de opcode y 4 bits del estado actual)
- Una única ROM de 2^{10} x 20 = 20 Kbits
- Muchas combinaciones serán *don't care.* Mejores resultados usando más de una ROM.

UC con una Máquina de estados

Con un PLA

- Codifica únicamente los términos producto necesarios.
- Reduce la cantidad de almacenamiento requerido para control.
- Tamaño proporcional a: (#entradas + #salidas) x #terminos producto

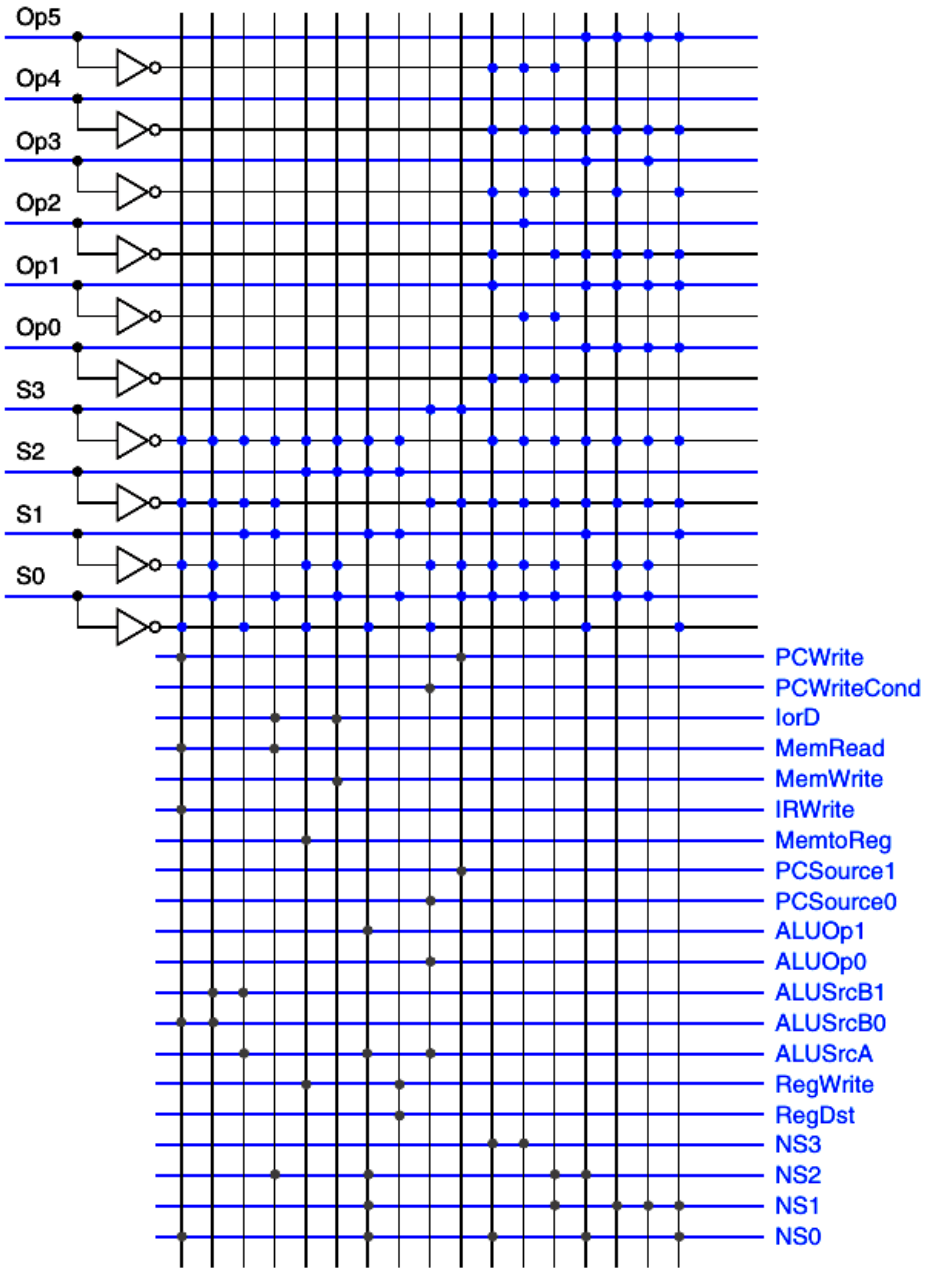


UC con una Máquina de estados

Con un PLA

- 10 entradas
  20 salidas
  17 términos producto

  Tamaño:
  (10+20) 17 = 510

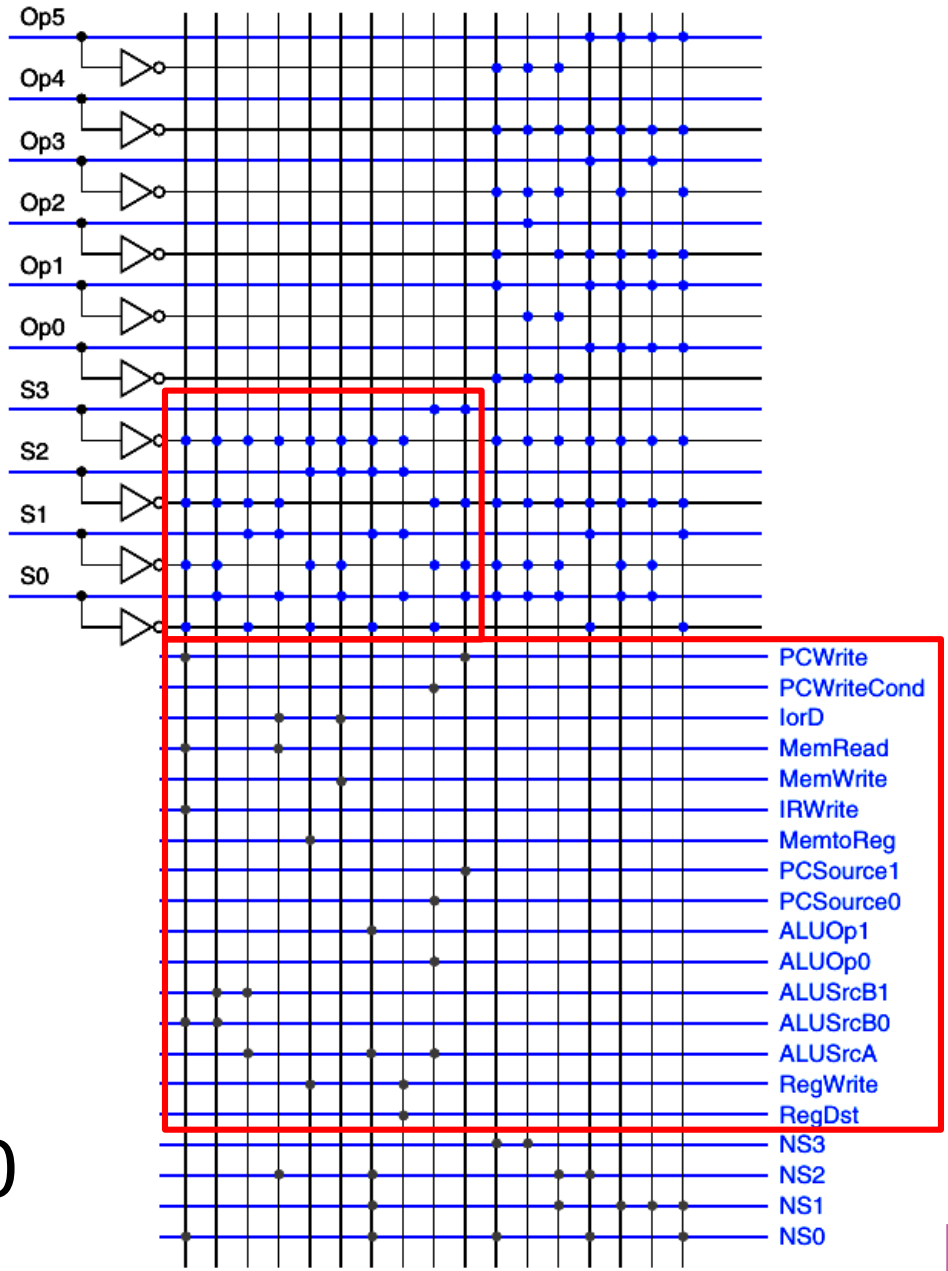


UC con una Máquina de estados

Tamaño con 2 PLAs

- 4 entradas
  16 salidas
  10 términos producto
  Tamaño:
  (4 + 16) 10 = 200

- 10 entradas
  4 salidas
  10 términos producto
  Tamaño: (10+4) 10 = 140

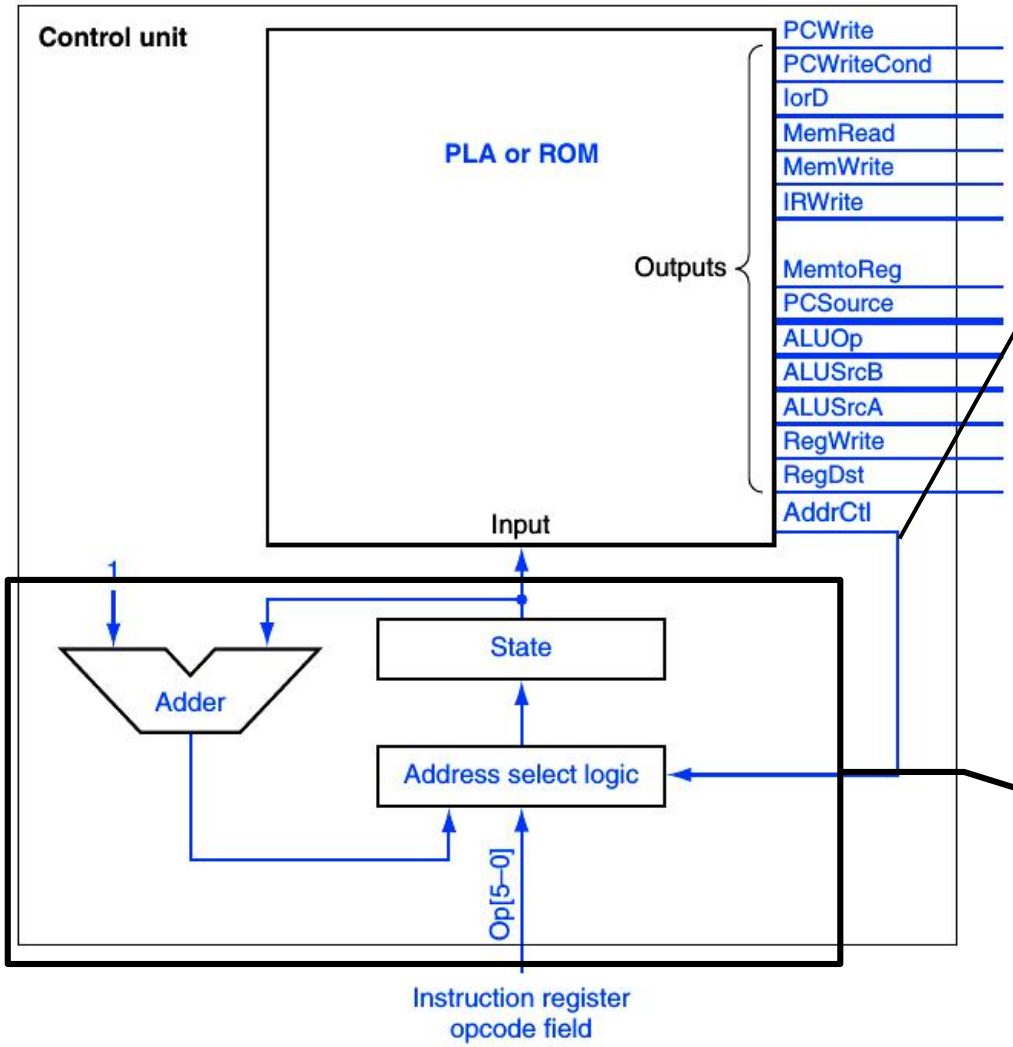


UC con una Máquina de estados

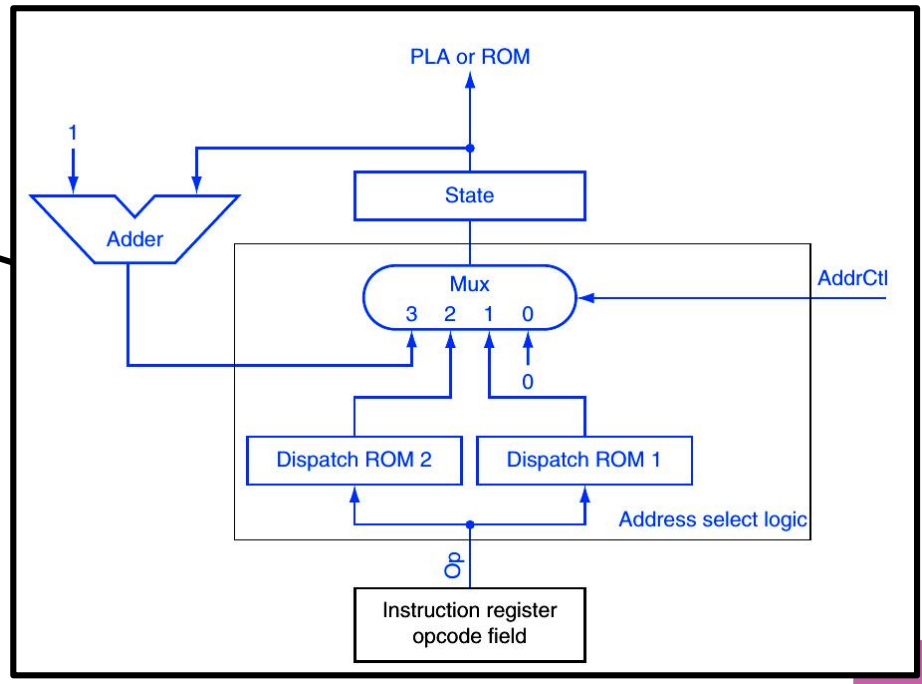
Con un contador

- Usa un contador para calcular el próximo estado
- Debe contemplan saltos a próximos estados no consecutivos:
  - Basándose en el opcode
  - Inicio próxima instrucción

UC con una Máquina de estados



AddrCtl	Acción
0	Estado 0
1	ROM 1
2	ROM 2
3	Estado incrementado



Microprogramación

Microprogramación

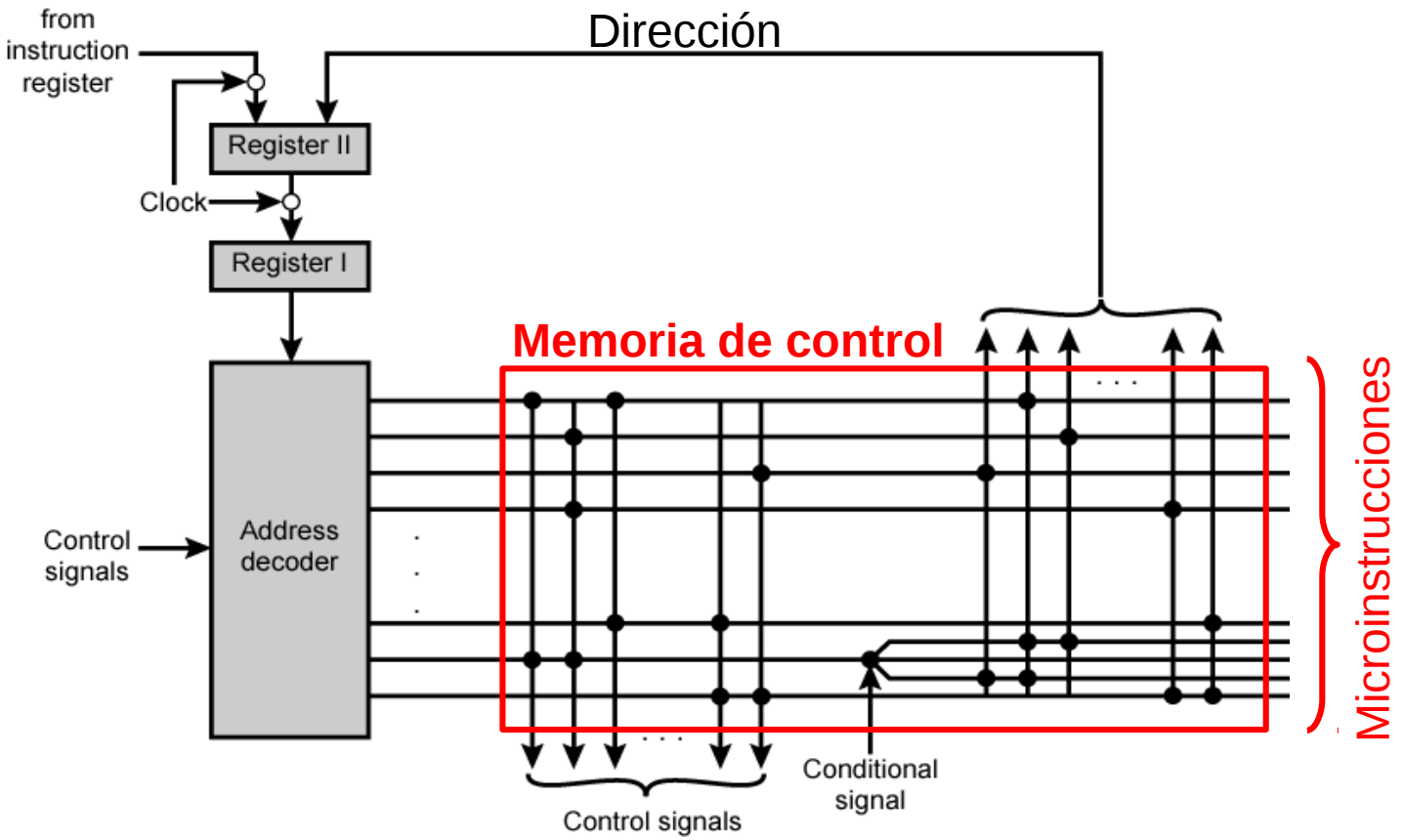
- La implementación de la UC como una máquina de estados usando un contador podría verse como una computadora.

- El concepto de microprogramación fue propuesto por M.V. Wilkes en 1951 y utilizaba un decoder y una ROM compuesta por una matriz de diodos.

Microprogramación

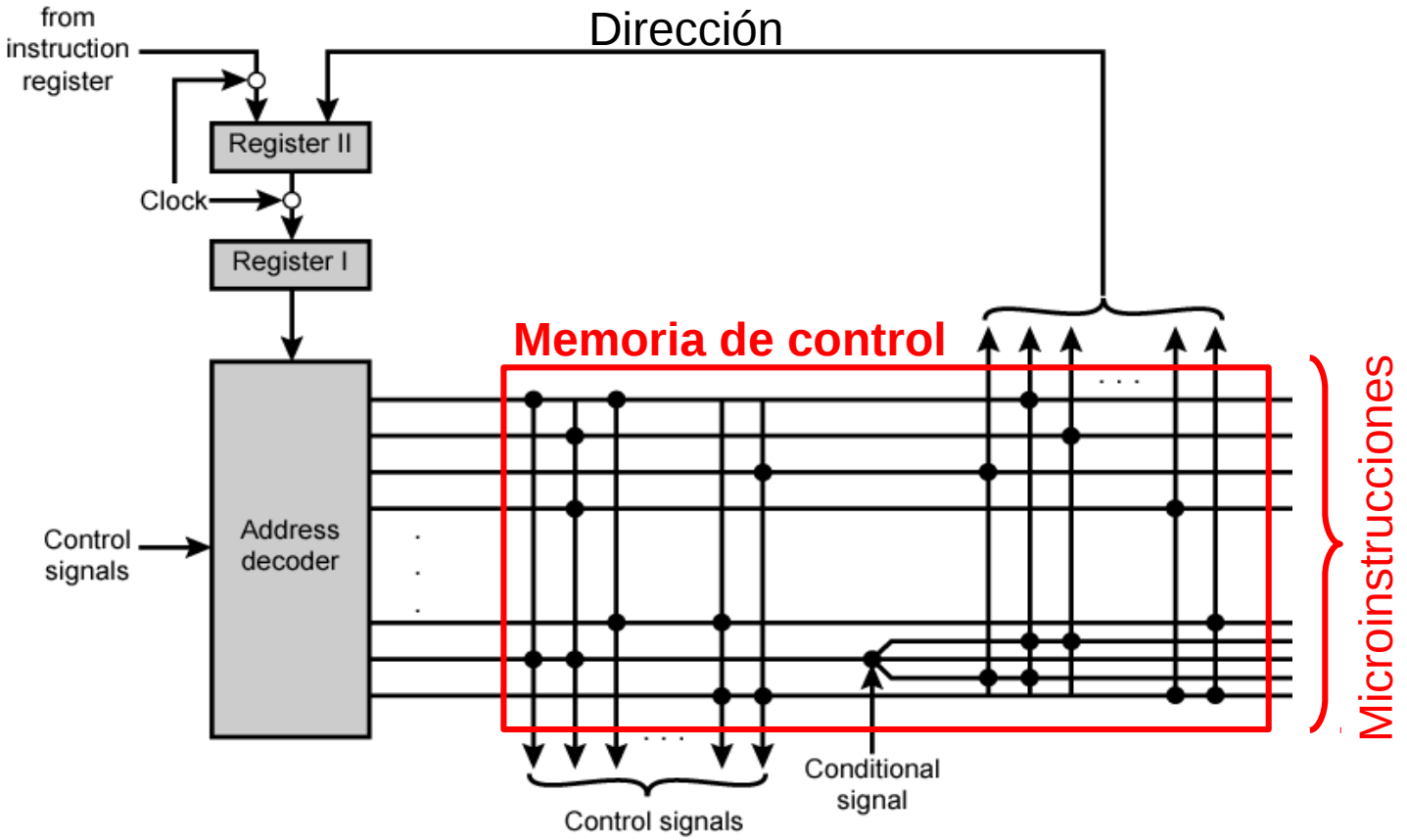
- *Microoperación*: Pasos elementales en los que se puede dividir la ejecución de una instrucción:
  - Incrementar el PC
  - Transferencia registro a registro
  - Suma en la ALU
  - ...
- *Microinstrucción:* conjunto de microoperaciones que ocurren en un momento de tiempo.
- *Microprograma (o firmware)*: Secuencia de microinstrucciones.

Microprogramación



- Durante un ciclo, se activa una fila de la matriz con un pulso.
- Genera las señales donde haya un diodo presente.
- La parte izquierda genera las señales de control.
- La parte derecha genera la próxima dirección.

Microprogramación



- La próxima dirección será el opcode del IR o la dirección proveniente de la matriz (dependiendo de las señales de control)

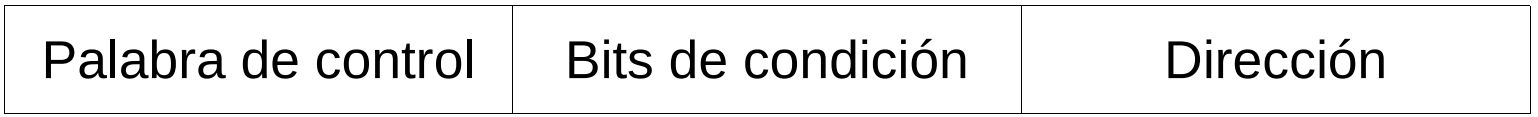
Microprogramación

- Por cada microoperación, la UC debe generar un conjunto predefinido de señales de control.
- Las señales de control pueden valer 1 (activada) o 0 (desactivada).
- Cada microoperación puede representarse como un patrón diferente de 1s y 0s.
- Este patrón se llama *palabra de control* (*control word).*
- Las *microinstrucciones* se organizan en una memoria.

Formato de la microinstrucción

Todas las microinstrucciones tiene tres campos importantes:

- *Palabra de control*
- *Dirección de la (posible) próxima microinstrucción*
- *Bits de condición*



Formato de la microinstrucción

La longitud de la microinstrucción está relacionada a:

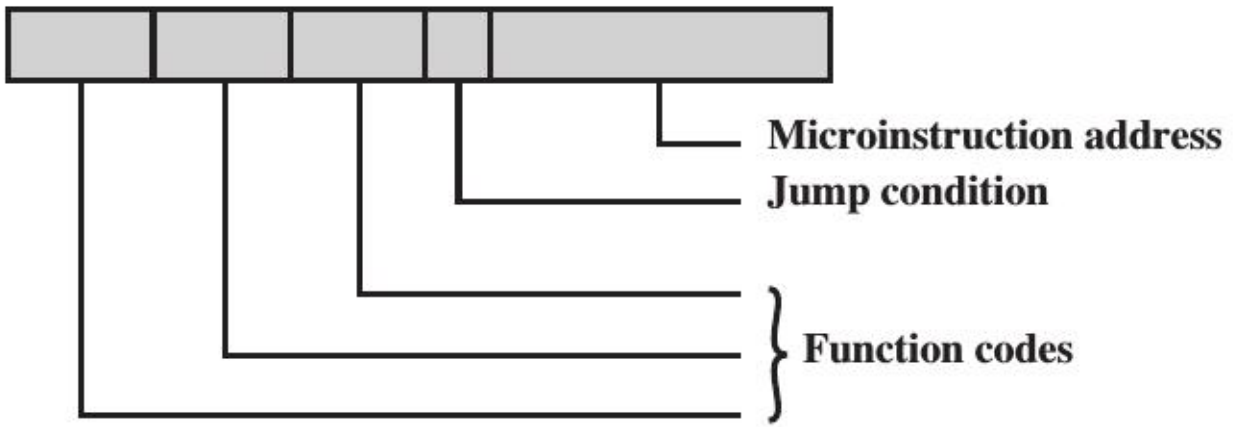
1) El número de microoperaciones que activa simultáneamente (grado de paralelismo)
2) El grado de codificación de la información de control.

En función del tamaño de la microinstrucción:

- Microprogramación vertical
- Microprogramación horizontal

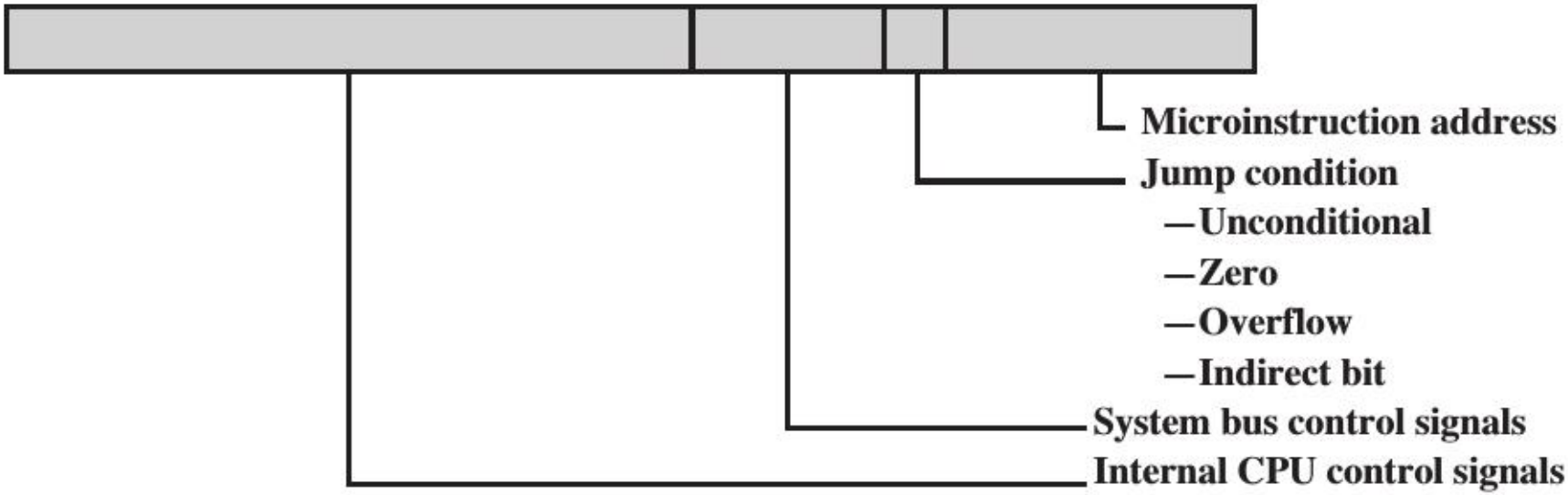
Microprogramación vertical

- Formato corto de microinstrucción.
- En cada ciclo puede activarse un número muy limitado de microoperaciones (x ej. 1).
- Alto grado de codificación de la información de control.



Microprogramación horizontal

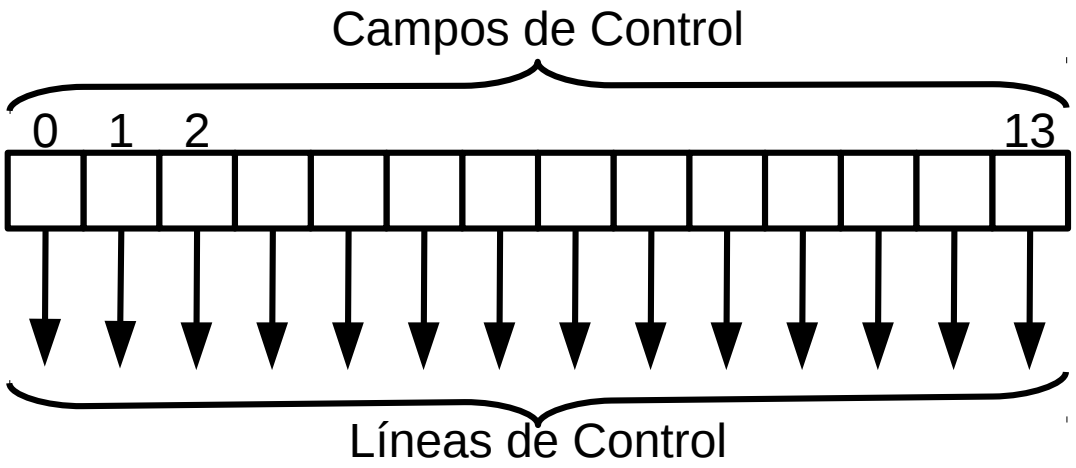
- Formato largo de microinstrucción.
- En cada ciclo puede activarse un gran número de microoperaciones.
- Poca codificación de la información de control.



El concepto de VLIW desciende de la microprogramación horizontal

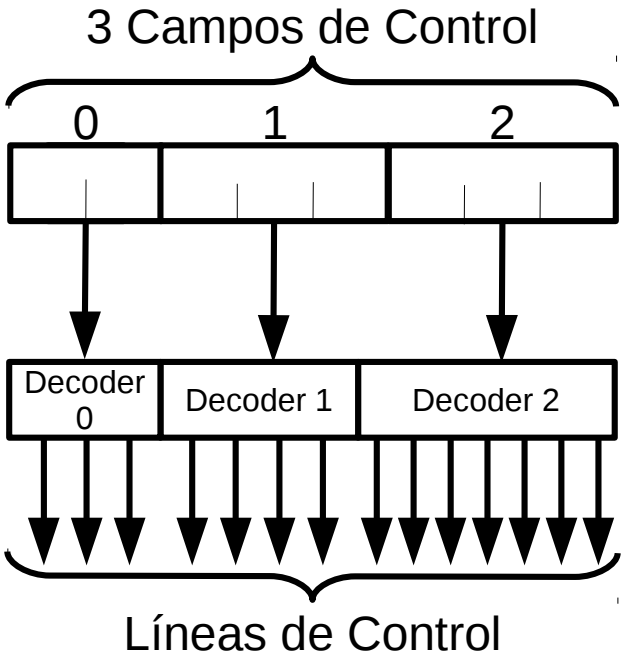
Codificación de la información de control

Sin codificación

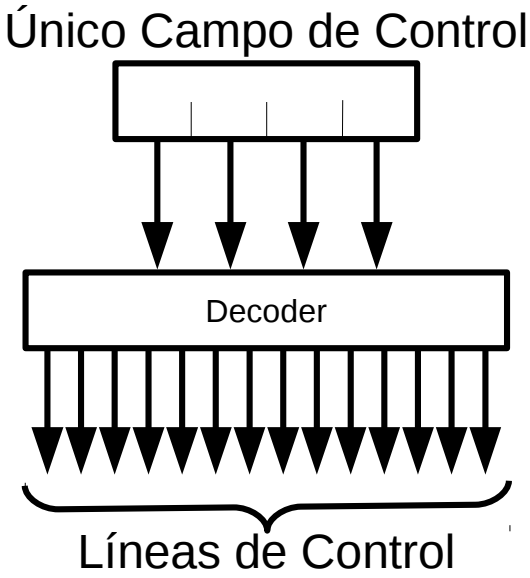


Codificación de la información de control

Codificación media



Codificación completa



Interpretación de la microinstrucción

1) Activar o desactivar las líneas de control siguiendo el patrón especificado por la palabra de control.

   Esto ejecuta una o más microoperaciones.

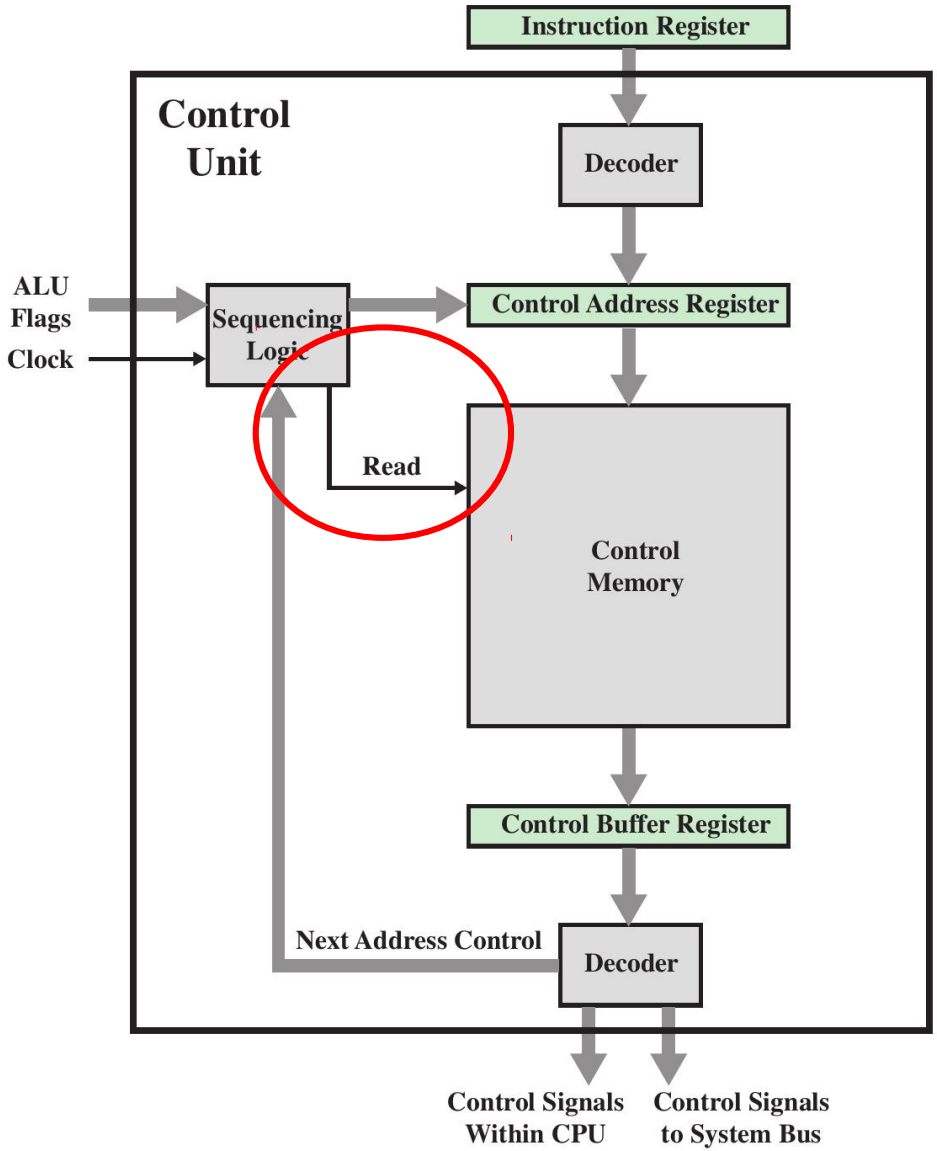
2) Si la condición indicada por los bits de condición es:

   i. Falsa: La próxima microinstrucción a ejecutar es la microinstrucción consecutiva.

   ii. Verdadera: La próxima microinstrucción a ejecutar es la indicada en el campo de dirección.

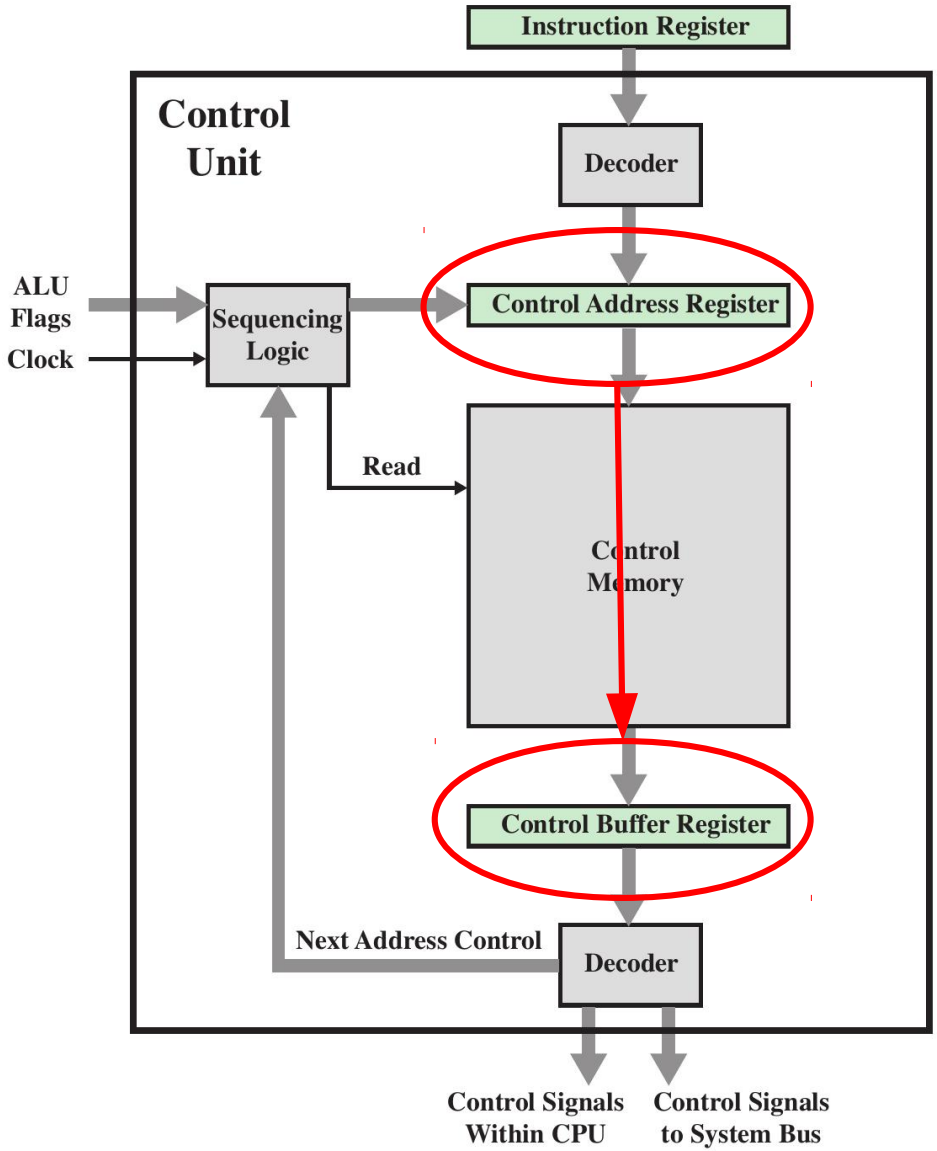
Funcionamiento

1) La lógica de secuenciamiento dispara una operación de lectura en la memoria de control.



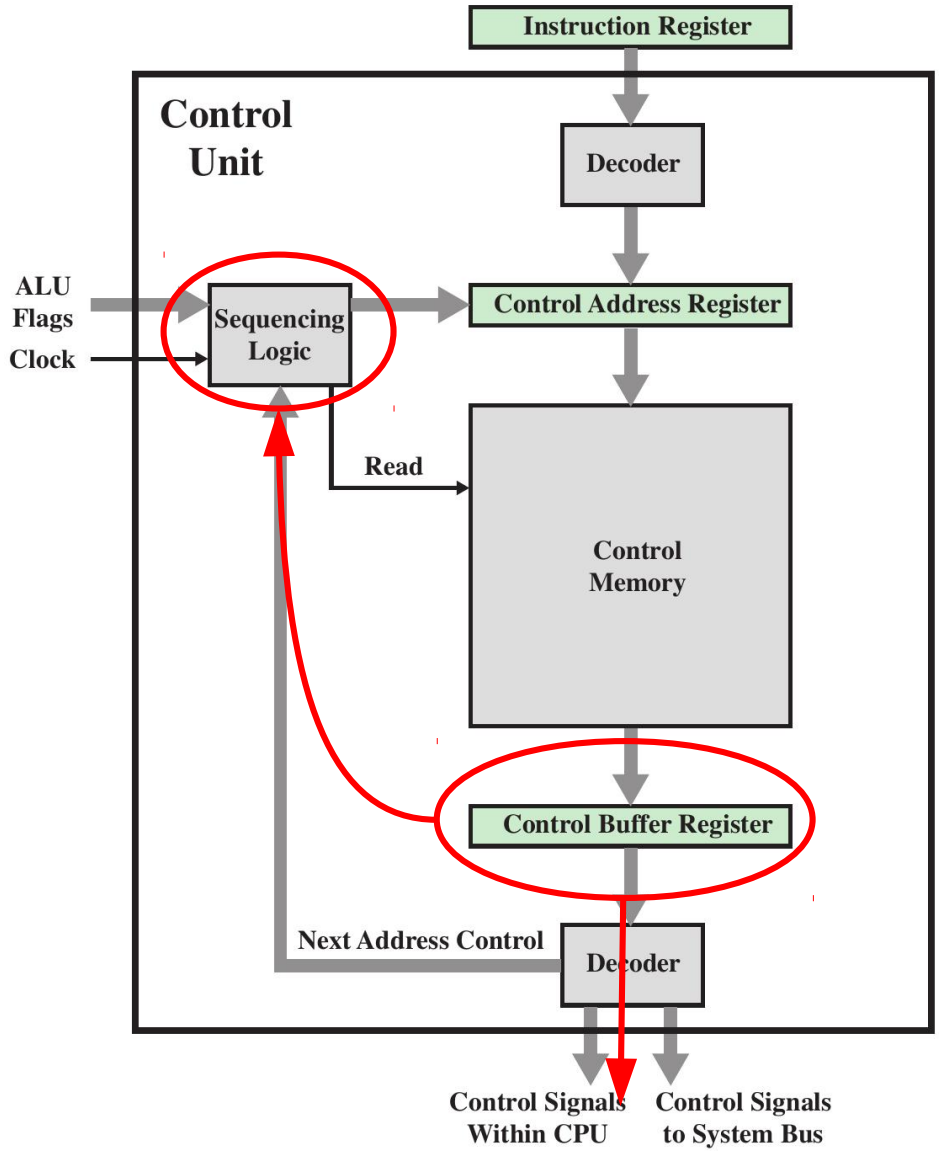
Funcionamiento

2) La palabra cuya dirección se especifica en el Control Address Register (CAR) se almacena en el Control Buffer Register (CBR)



Funcionamiento

3) El contenido del CBR genera señales de control e información sobre la próxima dirección.



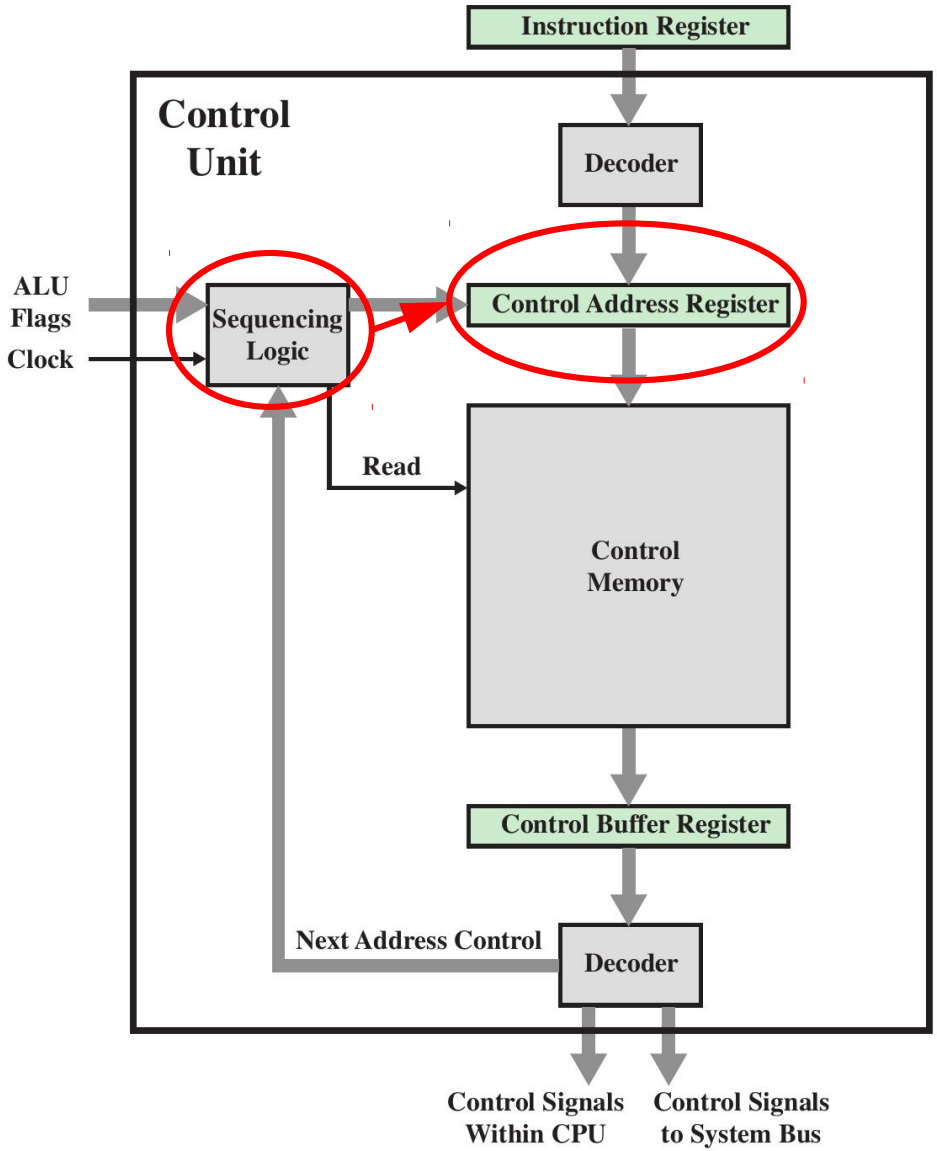
Funcionamiento

4) A partir de la información sobre la próxima instrucción del CBR y flags de la ALU, la lógica de control carga la próxima dirección en el CAR.

i. Cargar la instrucción consecutiva (+1 al CAR)

ii. Carga el campo de dirección del CBR en el CAR, basándose en la condición de salto de la microinstrucción.

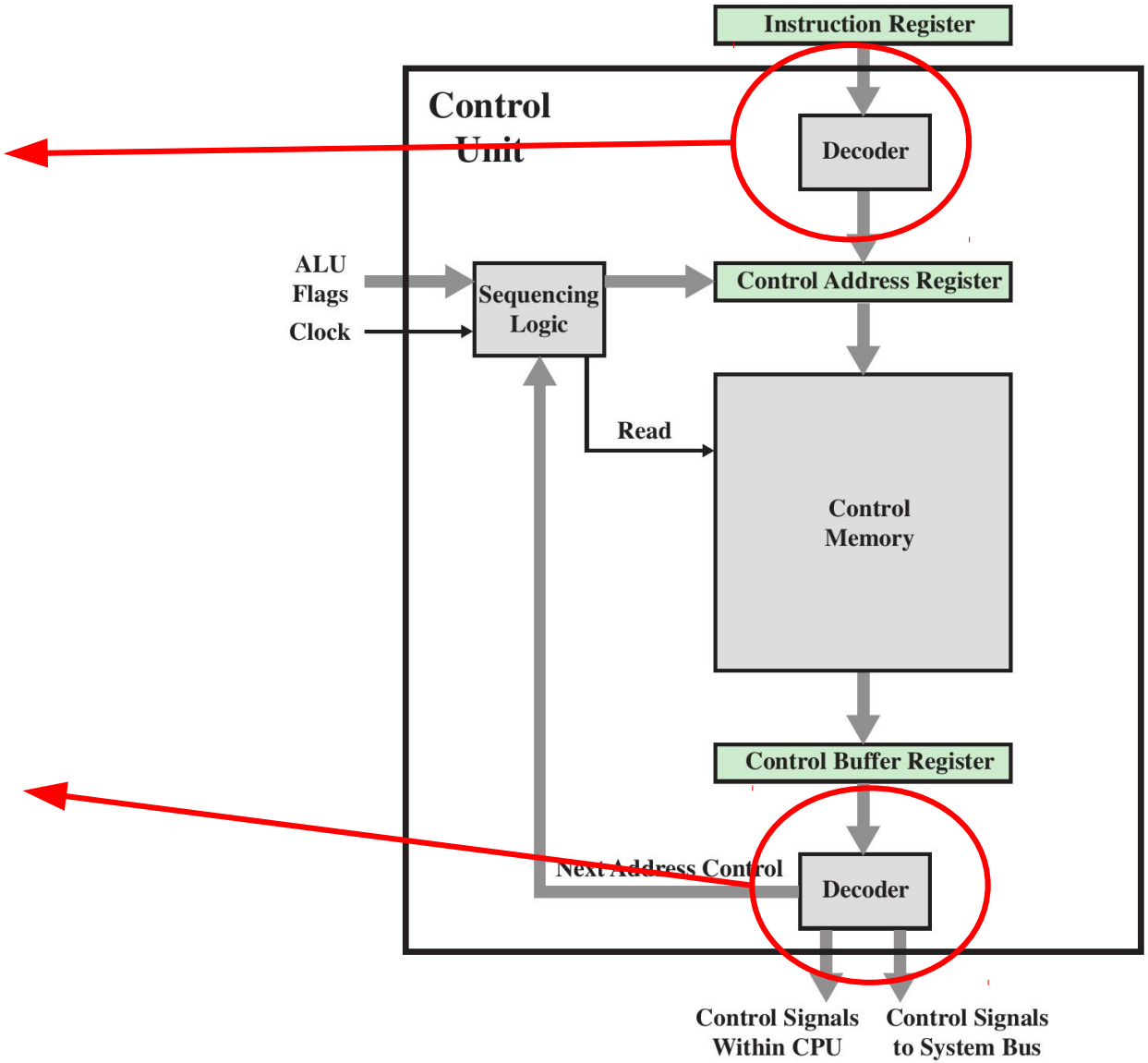
iii. Carga el CAR en función de del opcode del IR.



Funcionamiento

Decodifica el opcode del IR en una dirección de la memoria de control.

Decodifica los campos de control de la microinstrucción. Se usa solo en Microprogramación vertical.



Comparación

Ventajas y desventajas

Control cableado

– Lógica compleja para el secuenciamiento de las micro-operaciones

– Costoso de mantener y actualizar

+ Es más rápido

Dominante en arquitecturas RISC por el formato más simple de instrucciones.

Control microprogramado

+ Diseño simplificado

+ Más económico

+ Menos propenso a errores

– A tecnología comparable, es más lento.

Dominante en arquitecturas CISC.

Bibliografía

- *Apéndice D.* David A. Patterson & John L. Hennessy. *Computer Organization and Design. The Hardware/Software Interface.* Elsevier Inc. 2014, 5ta Ed.
- *Capítulo 7.* David Money Harris & Sarah L. Harris. Digital Design and Computer Architecture, Elsevier. 2013, 2da Ed.
- *Capítulos 19 y 20.* William Stallings. *Computer Organization and Architecture. Designing for Performance*. Pearson. 2013, 9na Ed.

Suplementaria

- J. Hennessy & D. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann Publishers INC. 2011, 5ta Ed.
- J. Hayes, *Computer Architecture and Organization*. McGraw-Hill 1978