Tema 3: II
Control Microprogramado

Tecnología de Computadores

Tema 4: Control Microprogramado

- Introducción
  - Bases del control microprogramado
- Unidad de control microprogramado del procesador r-MIPS multiciclo
  - Diseño básico de una U.C. microprogramada
  - Capacidad de salto
  - Microprogramación del conjunto de instrucciones

BASES DEL CONTROL MICROPROGRAMADO

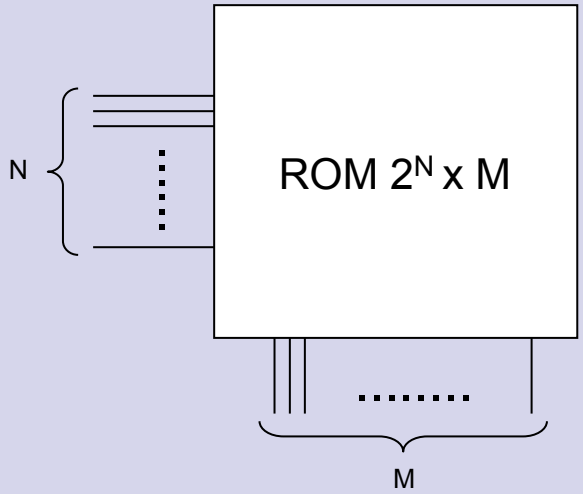
INTRODUCCIÓN

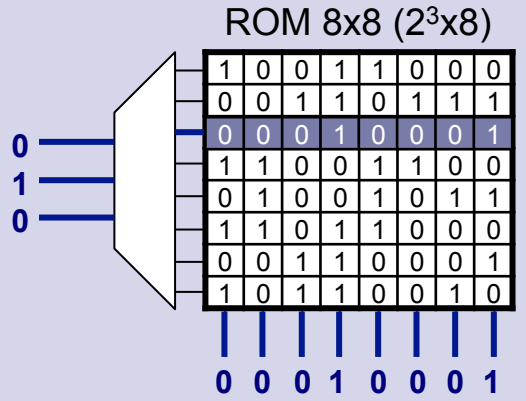
Introducción

- Alternativa al control cableado:

  Control Microprogramado

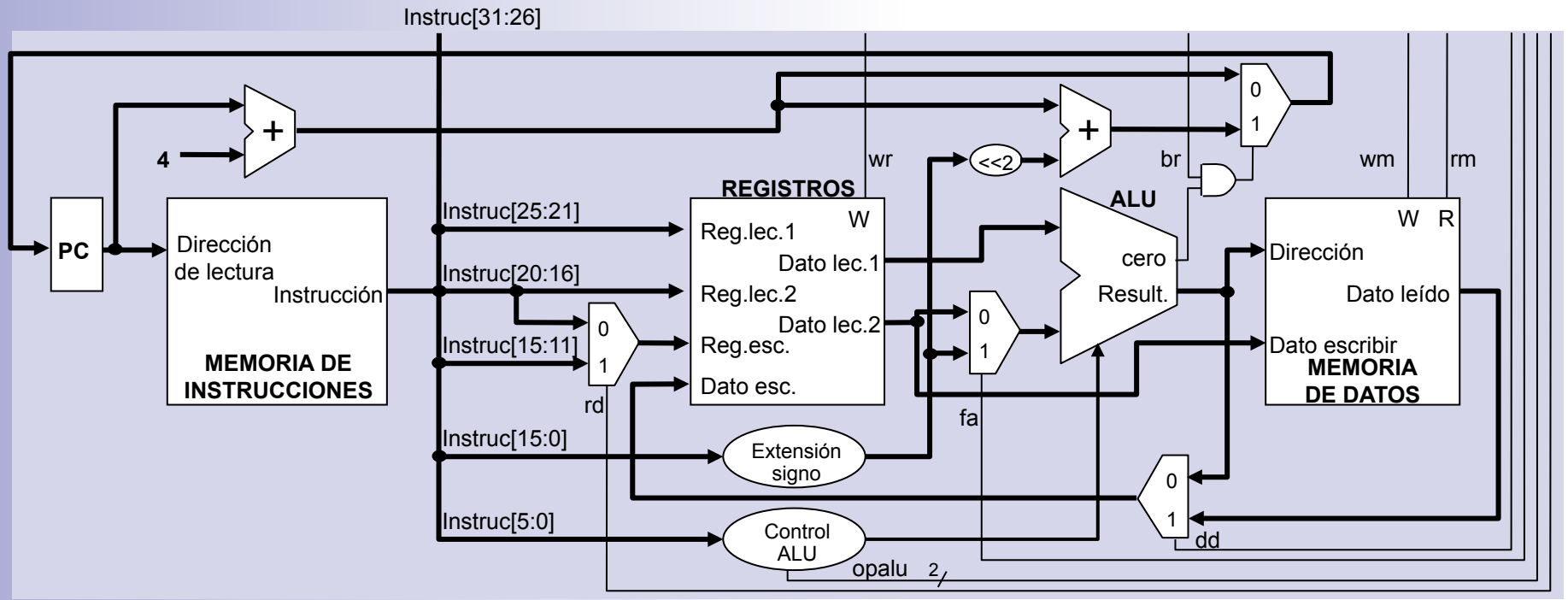
- Idea fundamental: Utilizar memoria ROM
  - La ROM almacenará el valor de las señales de control





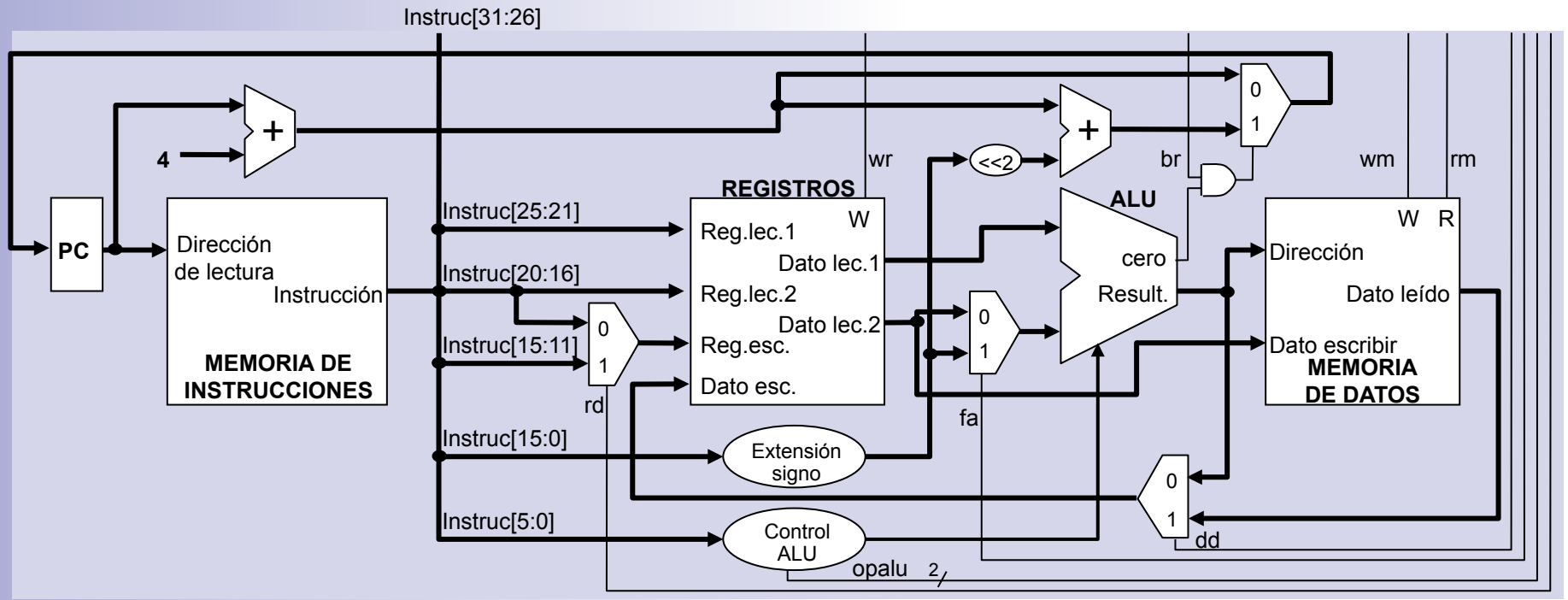
Introducción

- Caso más simple: U. Control Monociclo
  - Unidad Control = Circuito Combinacional
  - Función lógica para cada señal de control
    - Entradas: Código Instrucción
    - Salidas: Señales de control



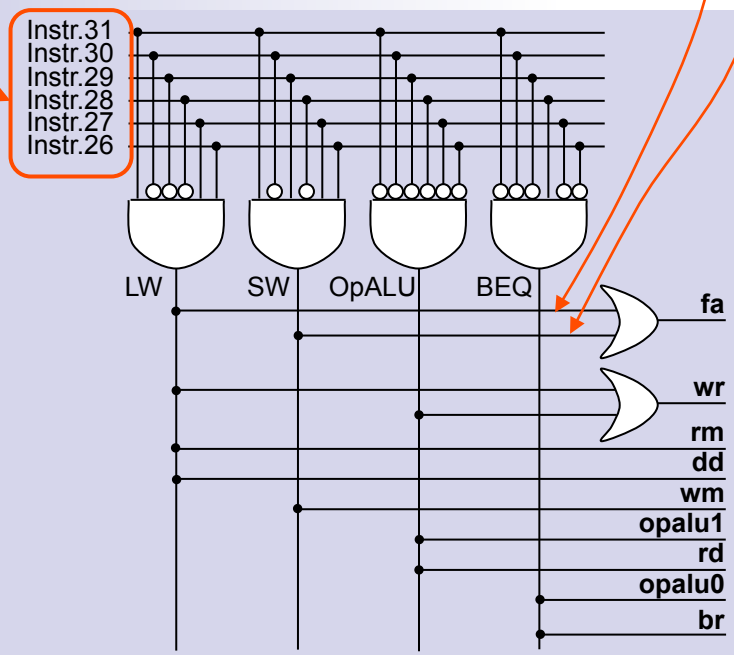
Introducción

Instrucción	Instr.[31:26]	Acción	Señales de control
LW rd,despl.(rf)	100011	rd ← MEM[rf + despl.]	fa, rm, dd, wr
SW rd,despl.(rf)	101011	MEM[rf + despl.] ← rd	fa, wm
OpALU rd, rf1, rf2	000000	rd ← rf1 (OpALU) rf2	opealu1, rd, wr
BEQ rf, rd, despl.	000100	si (rd=rf) PC ← PC + 4 + despl.	opealu0, br



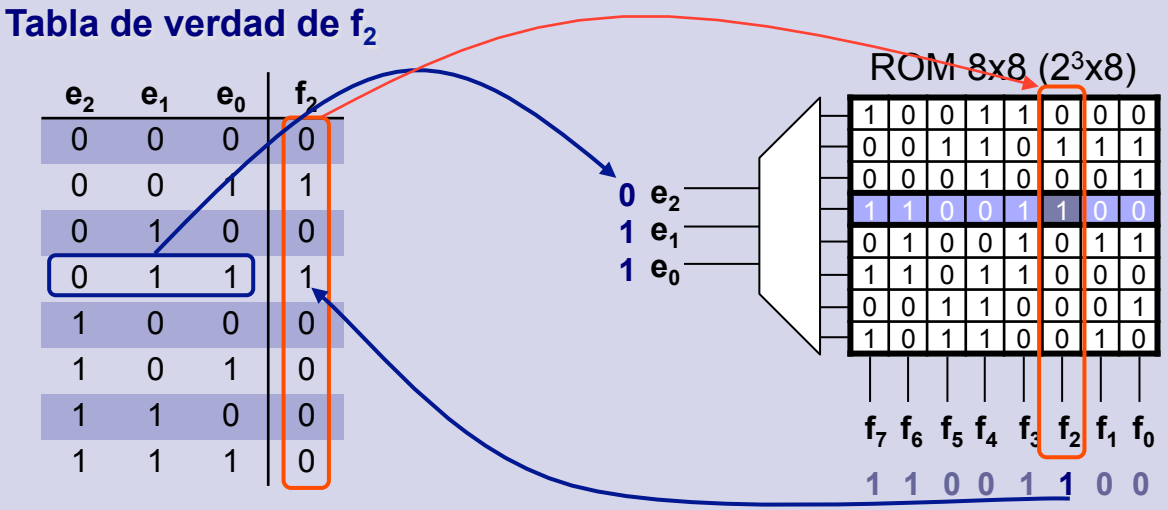
Introducción

Instrucción	Instr.[31:26]	Acción	Señales de control
LW rd,despl.(rf)	100011	rd ← MEM[rf + despl.]	fa, rm, dd, wr
SW rd,despl.(rf)	101011	MEM[rf + despl.] ← rd	fa, wm
OpALU rd, rf1, rf2	000000	rd ← rf1 (OpALU) rf2	opealu1, rd, wr
BEQ rf, rd, despl.	000100	si (rd=rf) PC ← PC + 4 + despl.	opealu0, br



Introducción

- Memoria ROM
  - Utilizar la ROM para "programar" la tabla de verdad
  - ROM 2^NxM ≡ $f_0(e_{N-1},\ldots,e_0),\ldots,f_{M-1}(e_{N-1},\ldots,e_0)$
    - Síntesis de M funciones de N bits
    - Combinación valores entrada = dirección de la ROM
    - $f_2(0,1,1)$ = bit 2 de la palabra 3 (011)

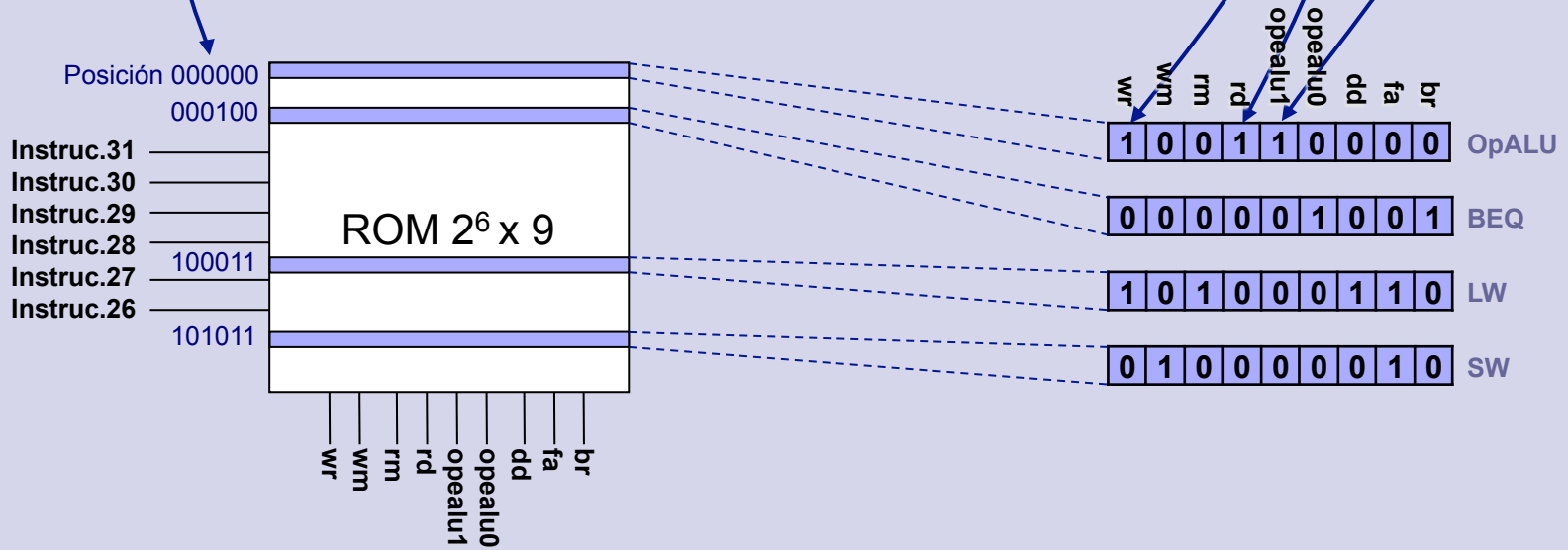


Introducción

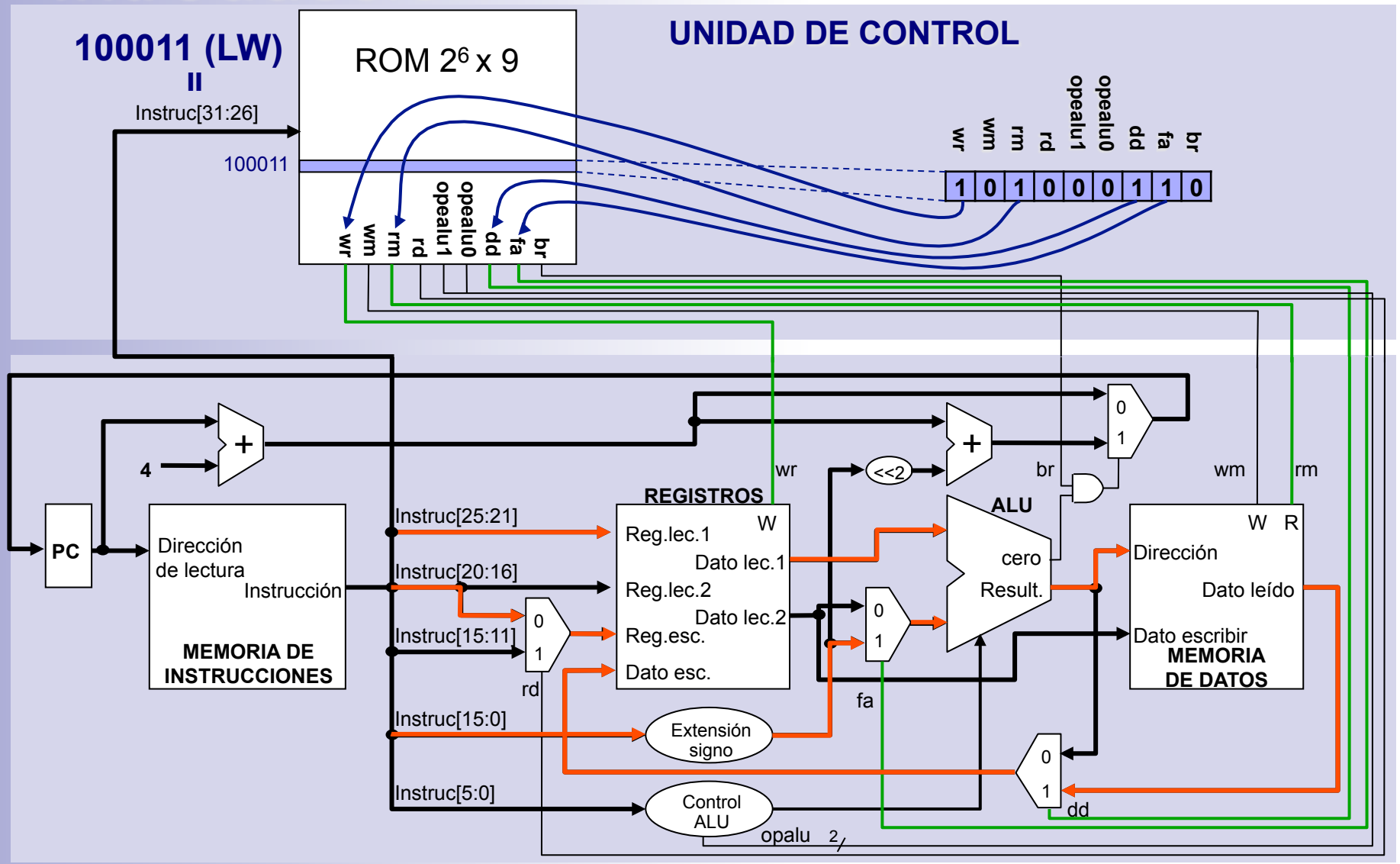
Instrucción	Instr.[31:26]	Acción	Señales de control
LW rd,despl.(rf)	100011	rd ← MEM[rf + despl.]	fa, rm, dd, wr
SW rd,despl.(rf)	101011	MEM[rf + despl.] ← rd	fa, wm
OpALU rd, rf1, rf2	000000	rd ← rf1 (OpALU) rf2	wr, rd, opealu1
BEQ rf, rd, despl.	000100	si (rd=rf) PC ← PC + 4 + despl.	opealu0, br

- Entrada = 6 bits (cód. instrucción)
- Salida = 9 bits (señales de control)

- **ROM 2^6 x 9**

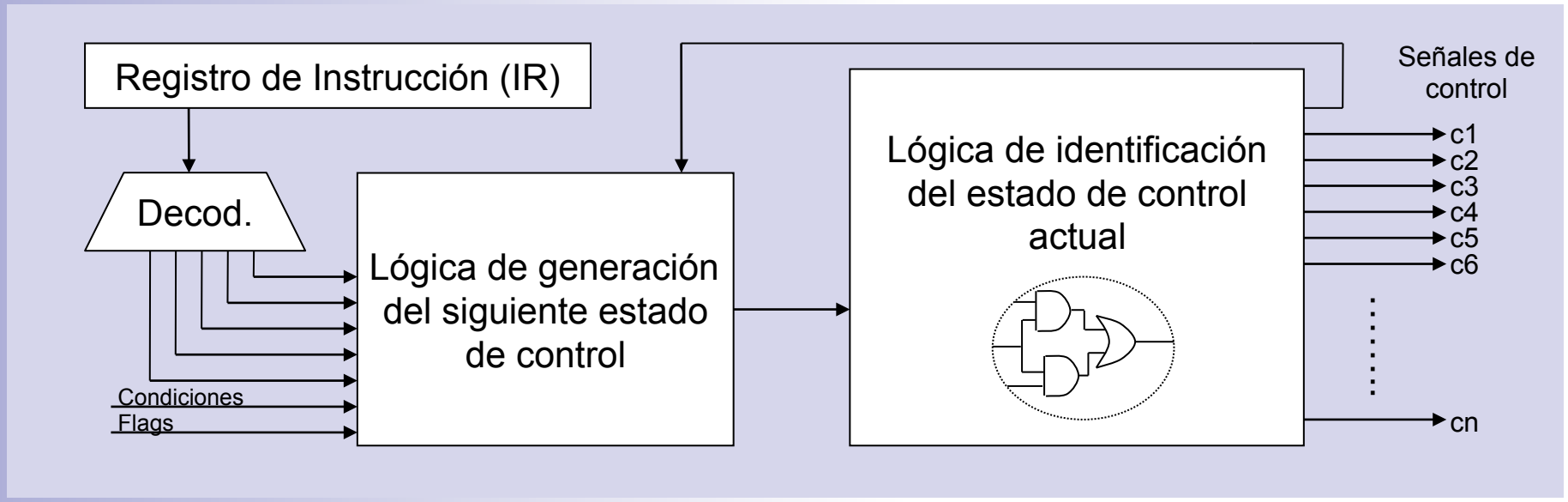


Introducción



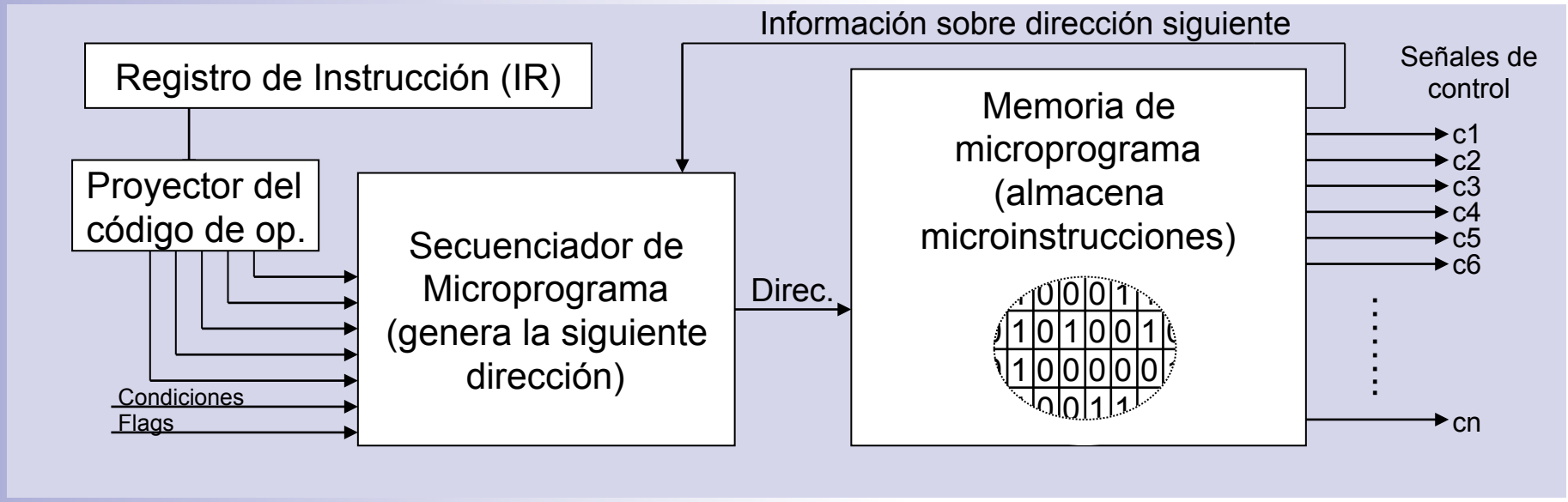
Introducción

- Unidad de control Multiciclo
  - Unidad de control = Circuito secuencial
    - Función lógica para cada señal de control
    - Lógica para generar secuencia de estados de control
    - Diseño heurístico, poco flexible y estructurado
    - De difícil mantenimiento y modificación (rediseño completo)



Introducción

- Unidad de control microprogramada
  - Idea: Bloque de memoria ROM
    - Almacena información estados de control
    - Estado de control actual identificado por dirección
    - Secuenciador de Microprograma
  - Diseño de la U.C. más simple y sistemático



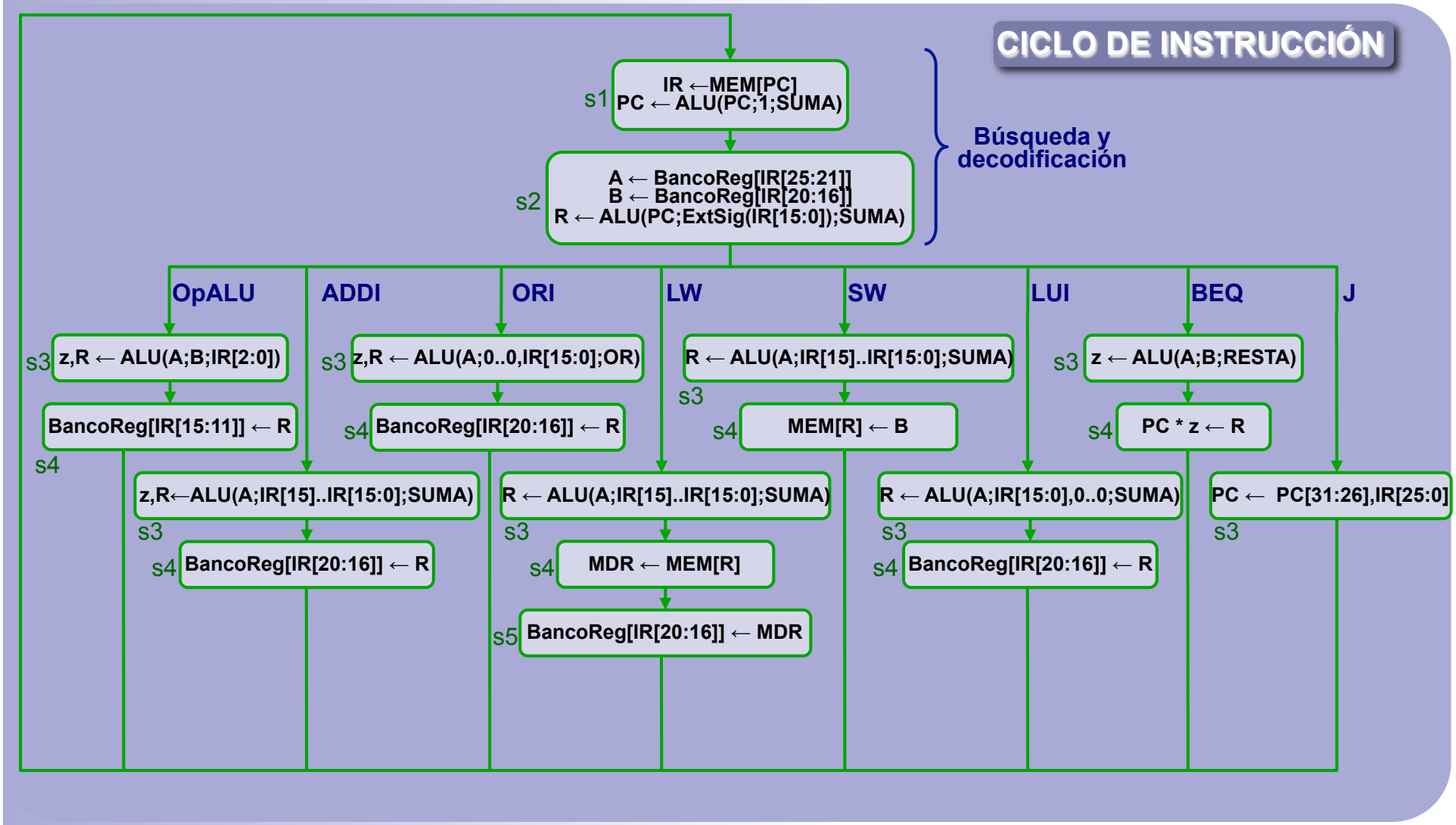
Tema 4: Control Microprogramado

- Introducción
  - Bases del control microprogramado
- Unidad de control microprogramado del procesador r-MIPS multiciclo
  - Diseño básico de una U.C. microprogramada
  - Capacidad de salto
  - Microprogramación del conjunto de instrucciones

DISEÑO BÁSICO DE LA U.C. MICROPROGRAMADA

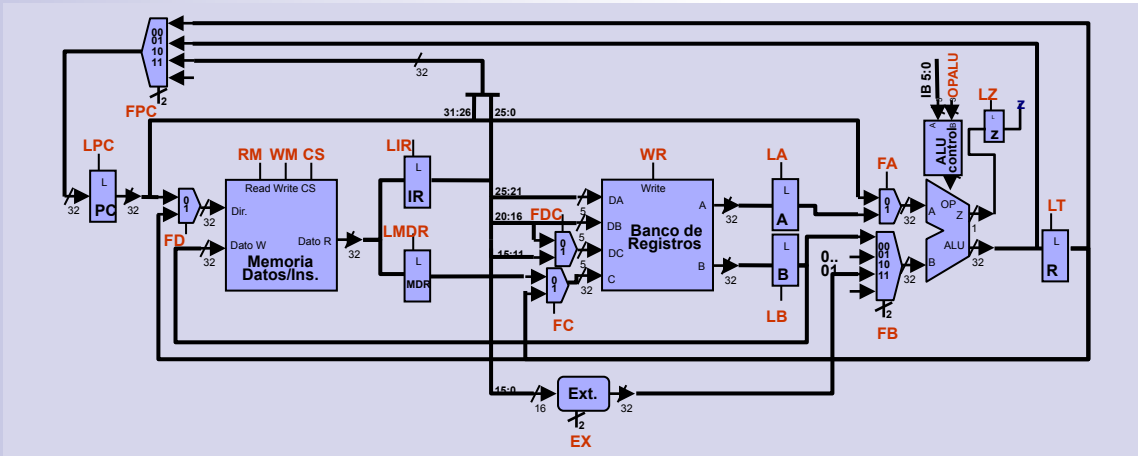
UNIDAD DE CONTROL MICROPROGRAMADO DEL PROCESADOR R-MIPS MULTICICLO

Diseño U.C. Microprogramada r-MIPS



Diseño U.C. Microprogramada

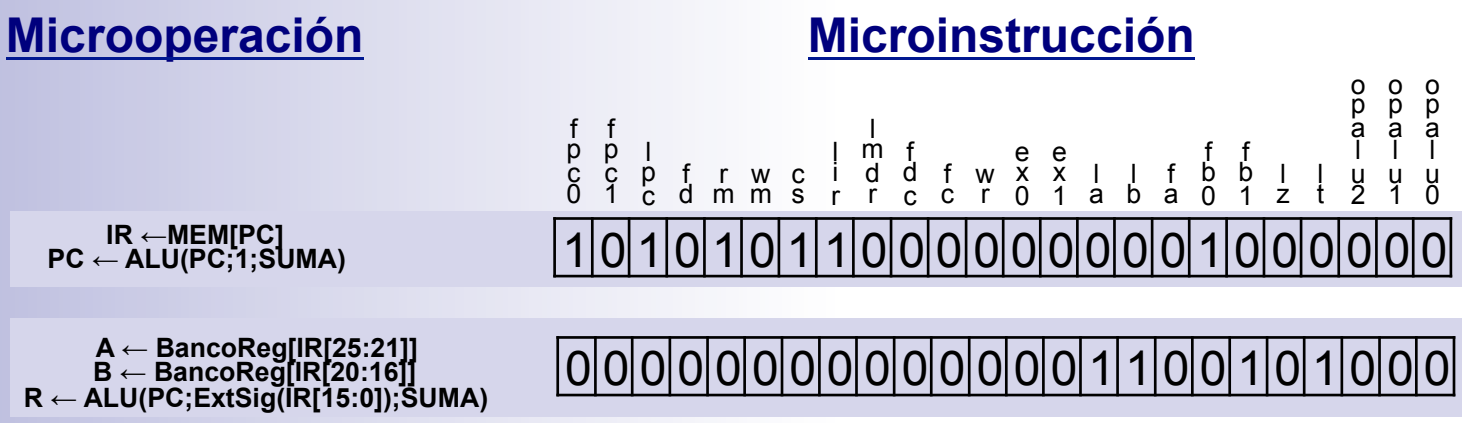
- Diseño básico: control mediante ROM
  - 1 bit de cada posición para cada señal de control
    - 24 señales de control = palabras de 24 bits
  - 1 posición de memoria por cada estado de control del ciclo de instrucción
    - 18 estados de control (2 (Búsqueda) + 2 (OpALU) + 2 (ADDI) + 2 (ORI) + 3 (LW) + 2 (SW) + 2 (LUI) + 2 (BEQ) + 1 (J))
  - ROM de al menos 18 posiciones de 24 bits
    - ROM 2^5 x 24 bits



Diseño básico de una U.C. microprogramada

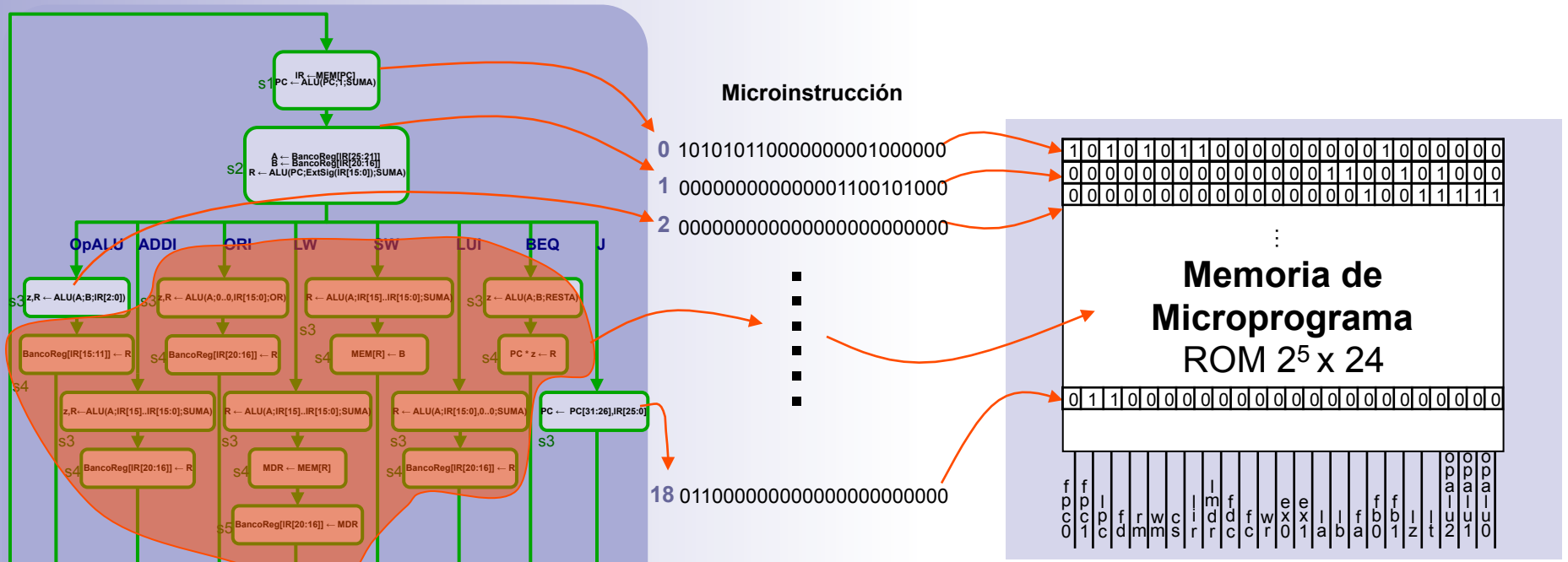
Diseño U.C. Microprogramada

- Diseño básico: Programación de microoperaciones individuales
  - Microoperación: transferencias a realizar en los estados de control
  - Microinstrucción: conjunto de 24 bits con 1 en posiciones de señales de control activas para realizar la microoperación



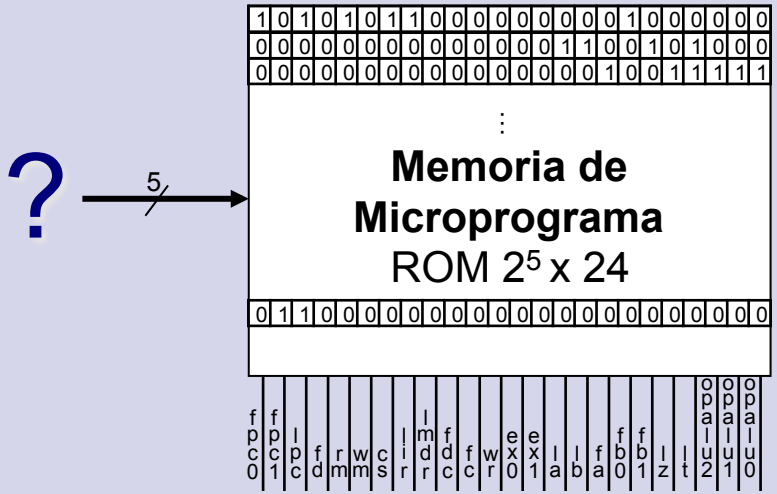
Diseño U.C. Microprogramada

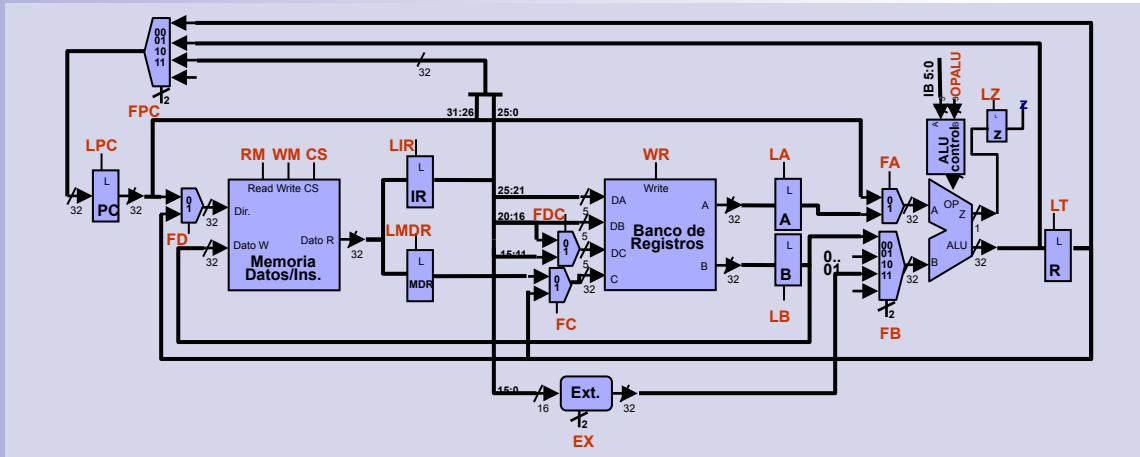
- Memoria de microprograma:
  - ROM donde se almacenan las microinstrucciones



Diseño U.C. Microprogramada

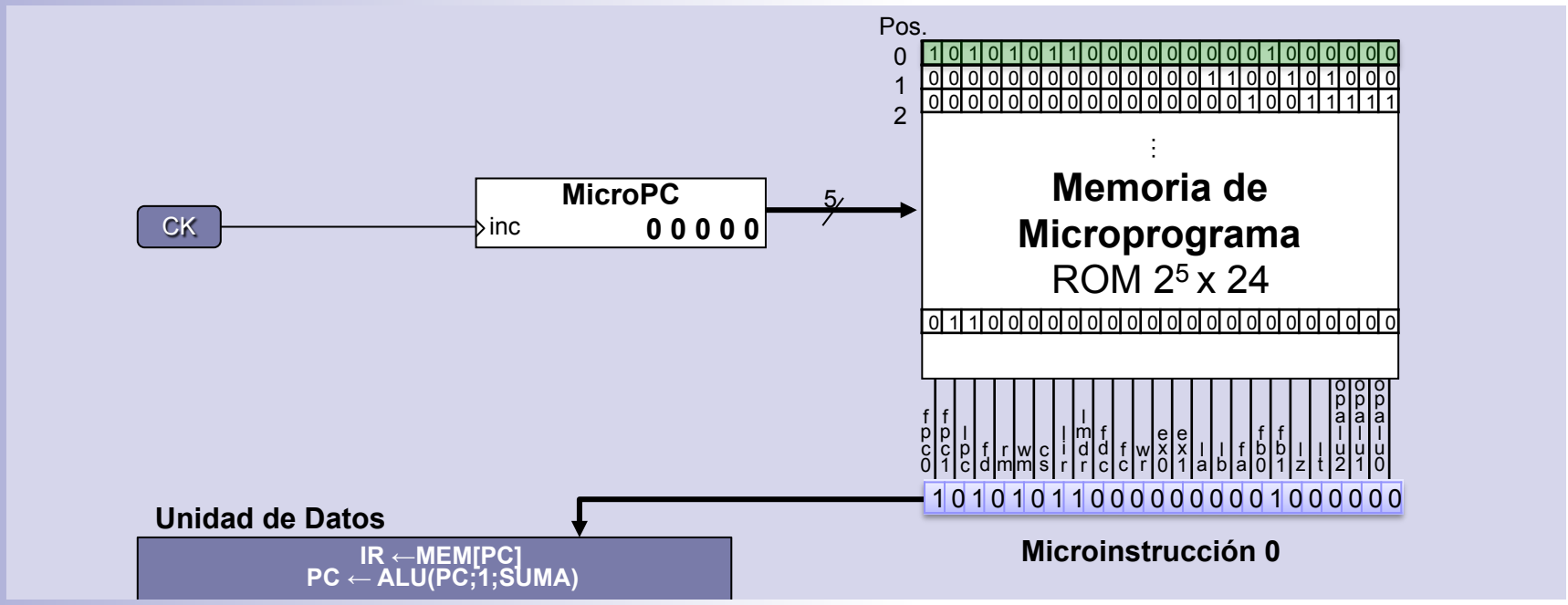
- Ejecución de una instrucción
  - Secuencia ordenada de microinstrucciones
- ¿Qué microinstrucción se ejecuta en cada instante?
  - Dir. ROM ≠ Código instrucción
- ¿Cuál es la siguiente microinstrucción?





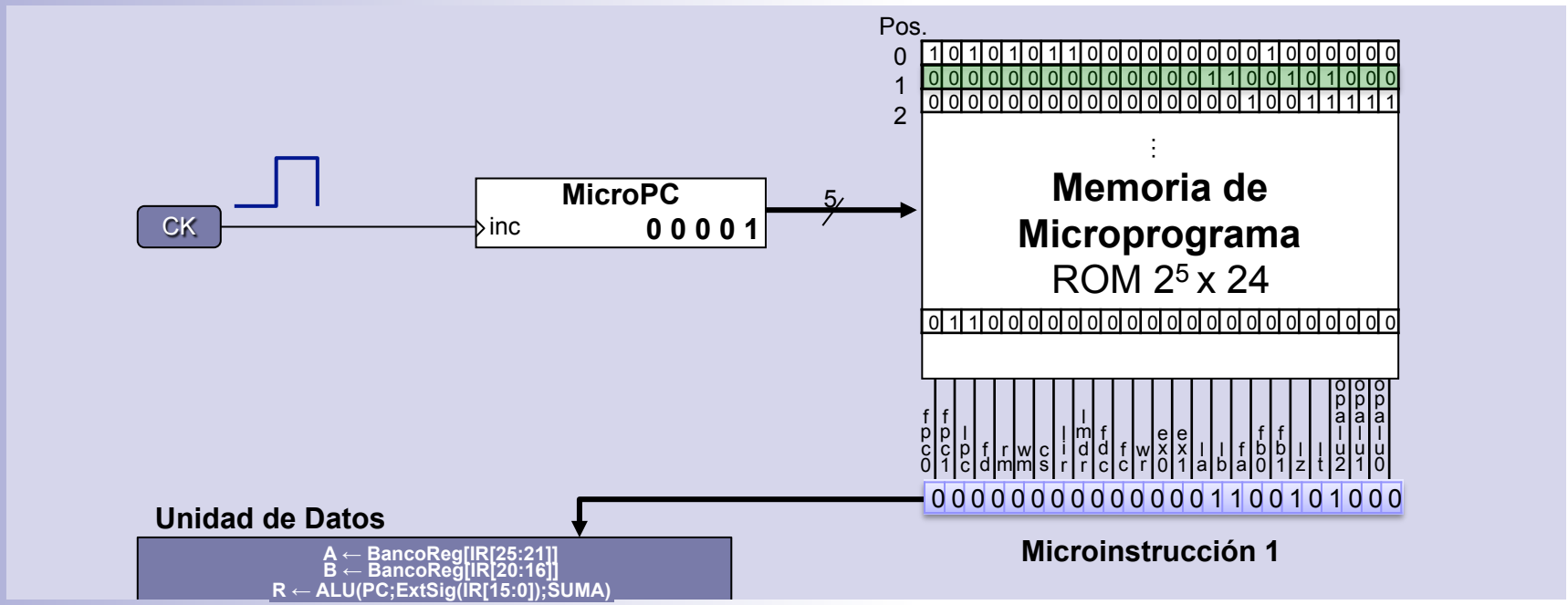
Diseño U.C. Microprogramada

- Secuenciación de microinstrucciones
  - Microinstrucciones ordenadas en ROM
  - Registro-contador para secuenciar
    - Contador de Microprograma (MicroPC)
    - Contiene dirección de la microinstrucción actual



Diseño U.C. Microprogramada

- Secuenciación de microinstrucciones
  - Microinstrucciones ordenadas en ROM
  - Registro-contador para secuenciar
    - Contador de Microprograma (MicroPC)
    - Contiene dirección de la microinstrucción actual

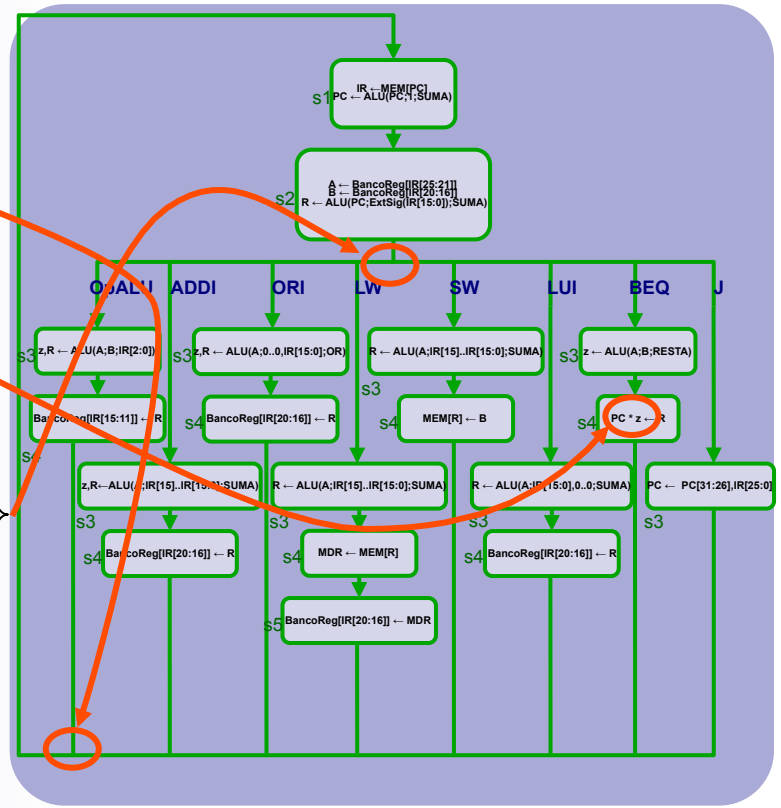


CAPACIDAD DE SALTO

UNIDAD DE CONTROL MICROPROGRAMADO DEL PROCESADOR R-MIPS MULTICICLO

Diseño U.C. Microprogramada

- Romper la lectura secuencial
  - La siguiente microinstrucción a ejecutar no está en la siguiente posición de la ROM
  - Ejemplos:
    - Última microinstrucción de una instrucción
    - Ejecución condicional de microinstrucciones
    - Última microinstrucción de la fase de búsqueda y decodificación

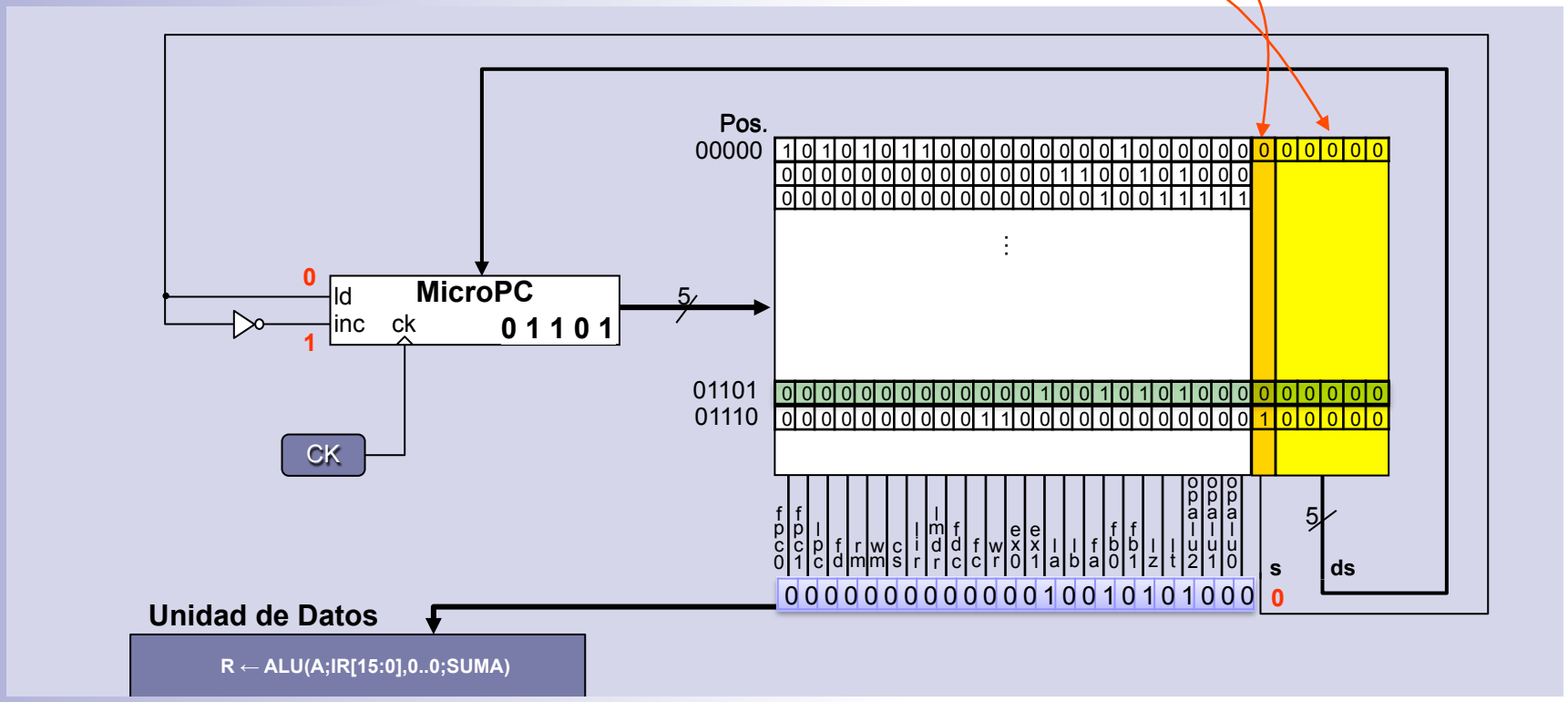


Diseño U.C. Microprogramada

- Capacidad de salto
  - Cargar MicroPC con una dirección diferente a la siguiente
  - Utilizar entrada de carga paralela del registro MicroPC
  - Modificación del formato de las microinstrucciones:
    - Añadir bit para decidir si se salta o no
    - Añadir conjunto de bits para especificar dirección de salto (a cargar en MicroPC)

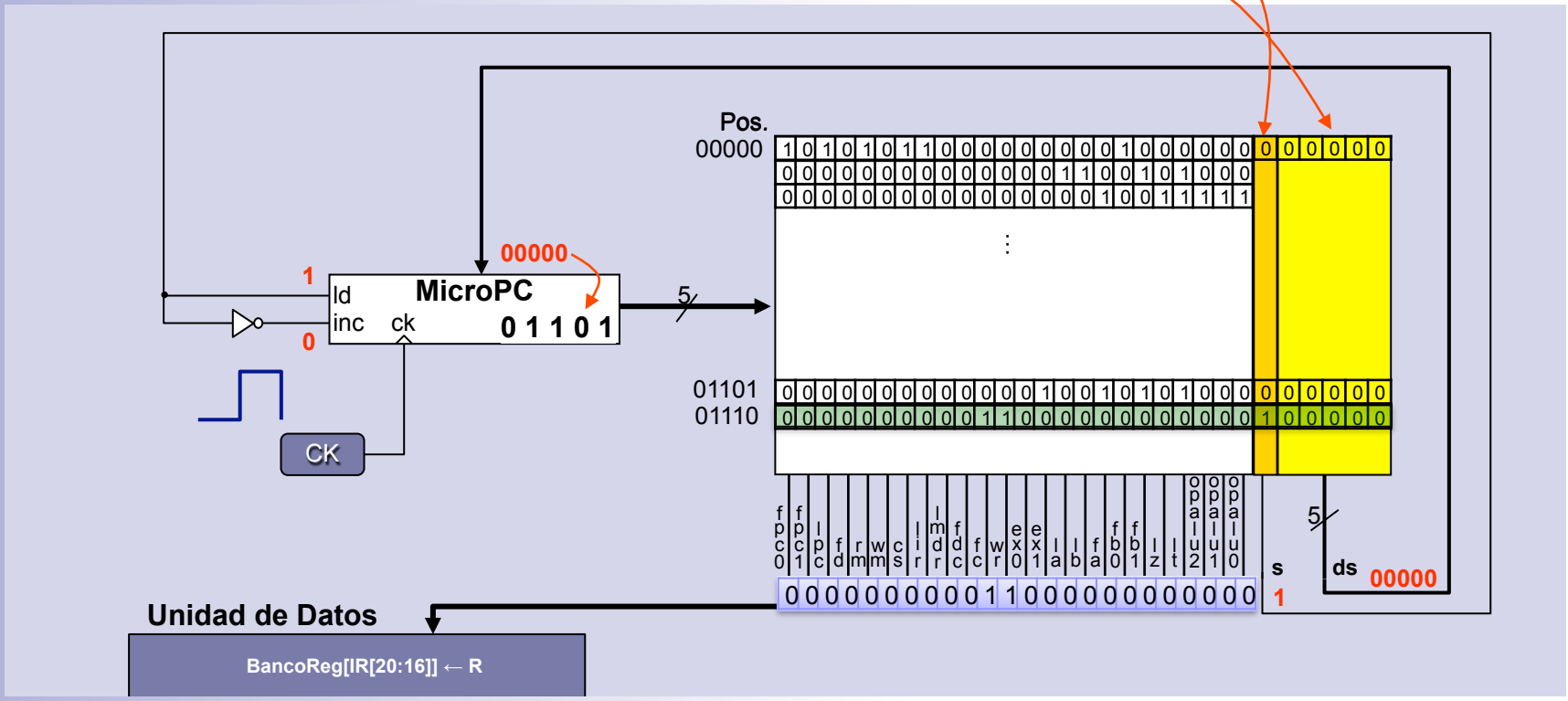
Diseño U.C. Microprogramada

- Modificación de formato de microinstrucción
  - Microinstrucción 14 - fin instrucción **LUI**
  - Siguiente microinstrucción: microinstrucción 0 (búsqueda y decodificación)
  - Añadir bit "**s**" de salto: 0 no salto, 1 salto
  - Añadir 5 bits "**ds**" de dirección de salto



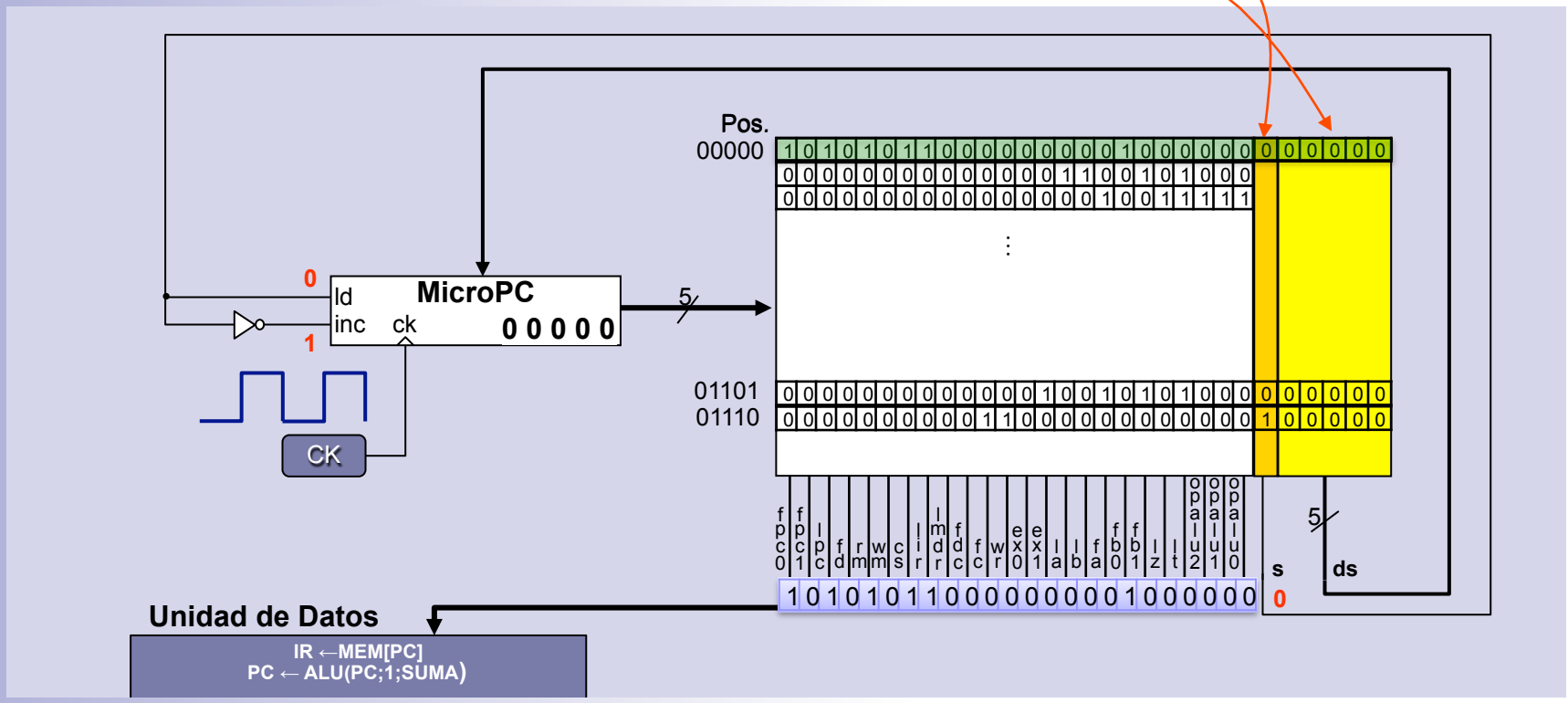
Diseño U.C. Microprogramada

- Modificación de formato de microinstrucción
  - Microinstrucción 14 - fin instrucción **LUI**
  - Siguiente microinstrucción: microinstrucción 0 (búsqueda y decodificación)
  - Añadir bit "**s**" de salto: 0 no salto, 1 salto
  - Añadir 5 bits "**ds**" de dirección de salto



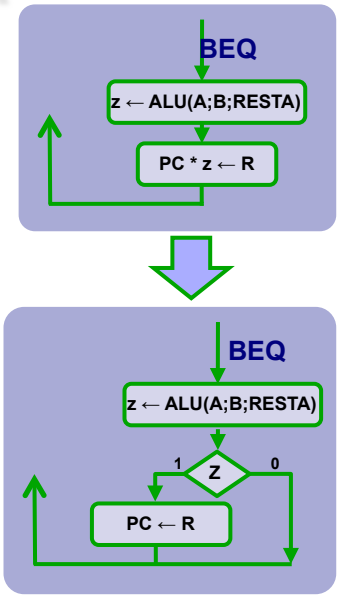
Diseño U.C. Microprogramada

- Modificación de formato de microinstrucción
  - Microinstrucción 14 - fin instrucción **LUI**
  - Siguiente microinstrucción: microinstrucción 0 (búsqueda y decodificación)
  - Añadir bit "**s**" de salto: 0 no salto, 1 salto
  - Añadir 5 bits "**ds**" de dirección de salto



Diseño U.C. Microprogramada

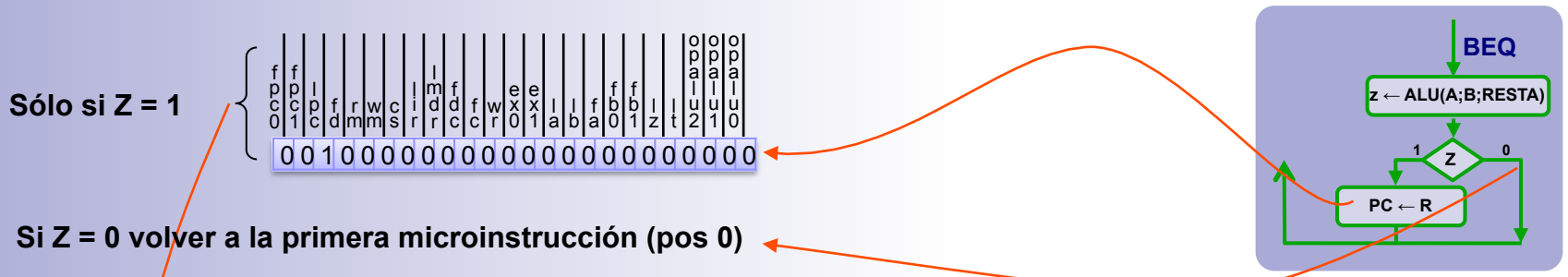
- Capacidad de salto condicional
  - Cargar MicroPC sólo si se cumple una determinada condición
  - Añadir lógica para la carga condicional de MicroPC
  - Modificación del formato de las microinstrucciones:
    - Añadir bits para decidir condición de salto
  - Condiciones de salto asociadas con los flags de estado (Z)
    - Ejemplo condiciones: saltar si Z, saltar si no Z
    - Codificar bit de salto "s" junto con condiciones
    - Necesitamos 2 bits, "**cs**", para determinar tipo de salto



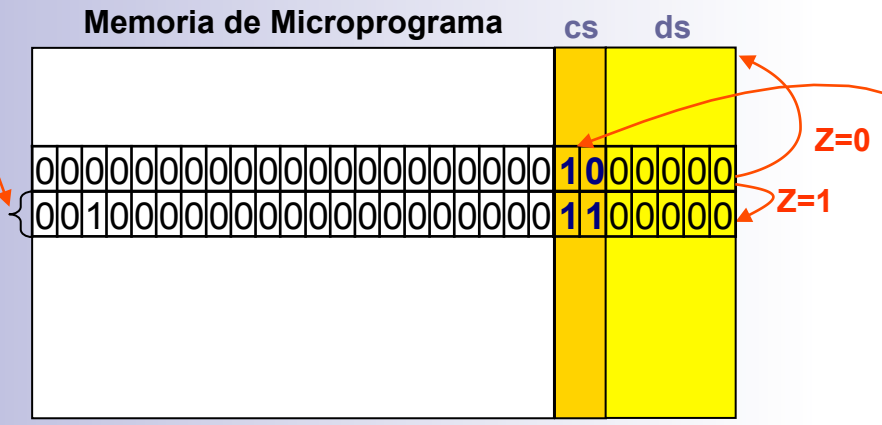
cs	Tipo salto
00	No saltar
01	Saltar si Z
10	Saltar si no Z
11	Salto incondicional

Diseño U.C. Microprogramada

- Microprogramación de BNZ utilizando salto condicional



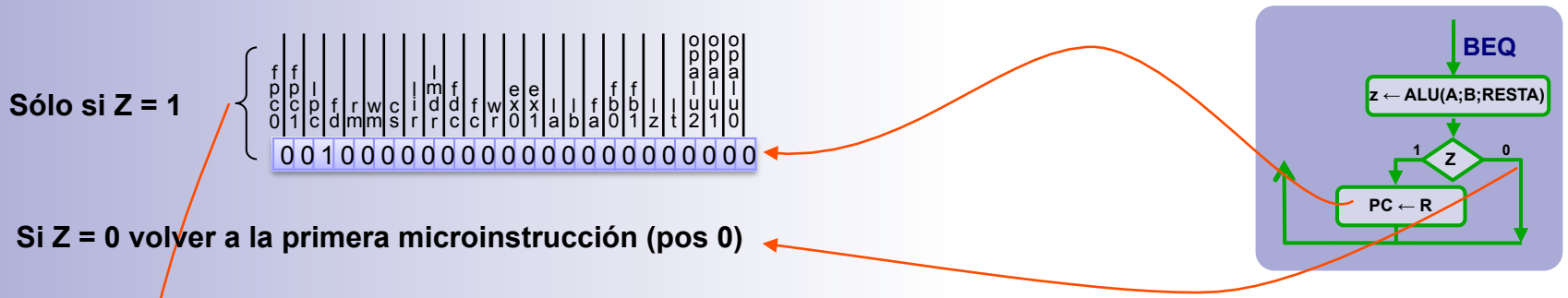
Introducir microinstrucción para determinar valor de Z



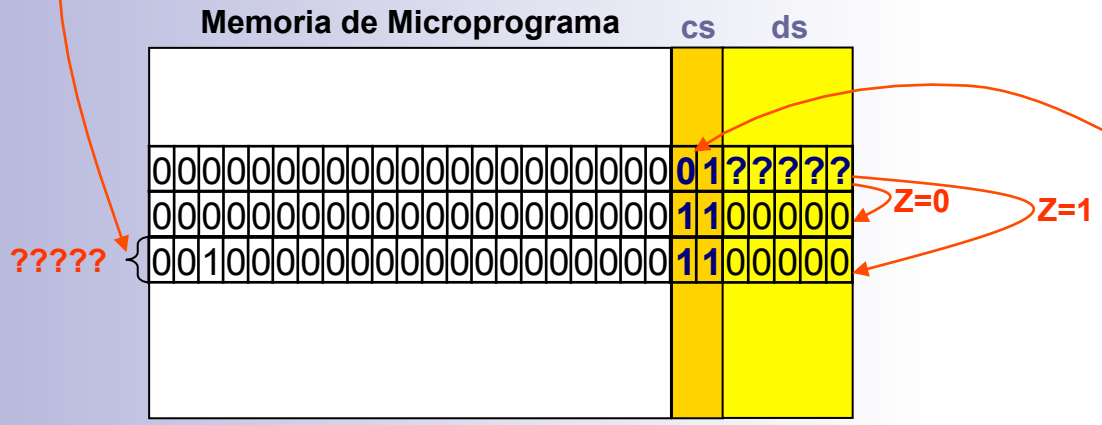
cs	Tipo salto
00	No saltar
01	Saltar si Z
10	Saltar si no Z
11	Salto incondicional

Diseño U.C. Microprogramada

- Microprogramación de BNZ utilizando salto condicional



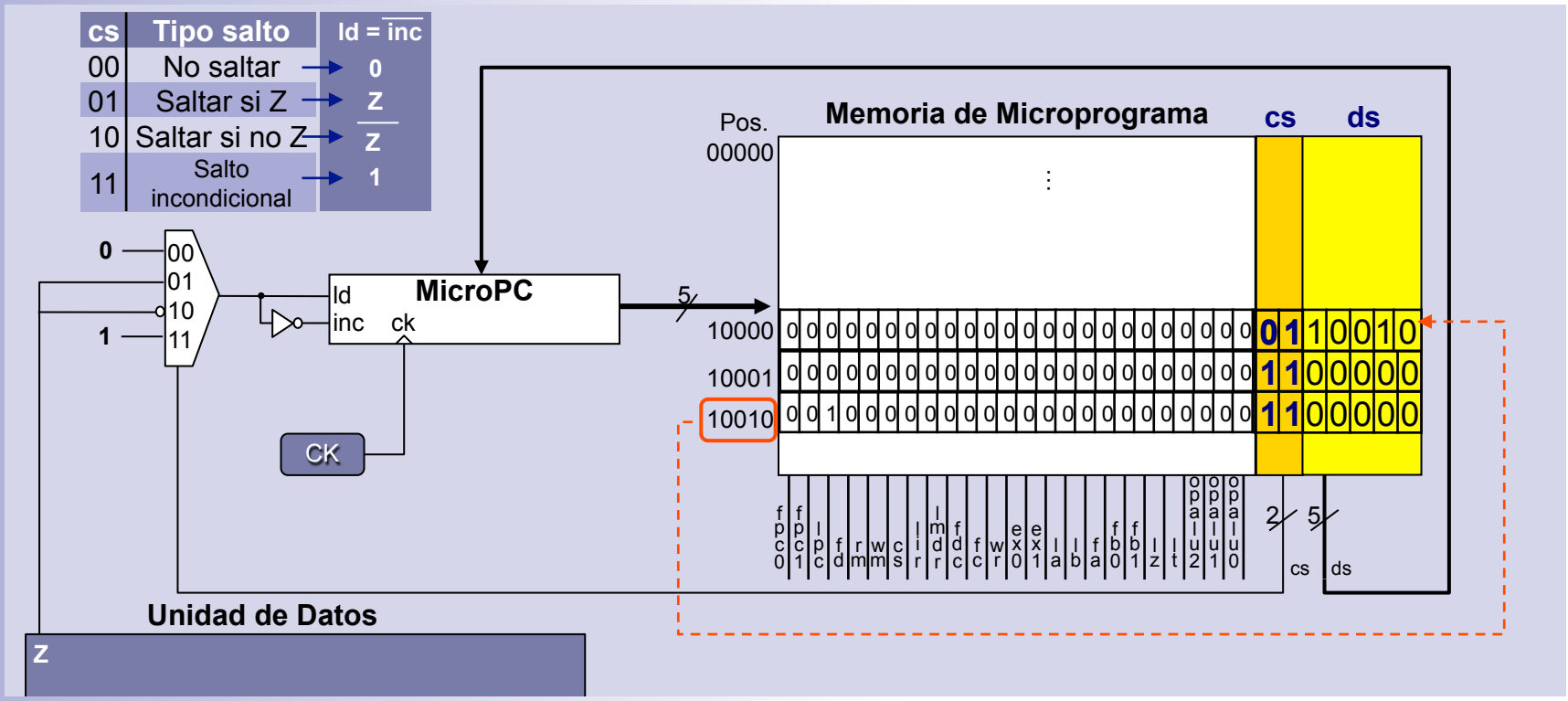
Introducir microinstrucción para determinar valor de Z



cs	Tipo salto
00	No saltar
01	Saltar si Z
10	Saltar si no Z
11	Salto incondicional

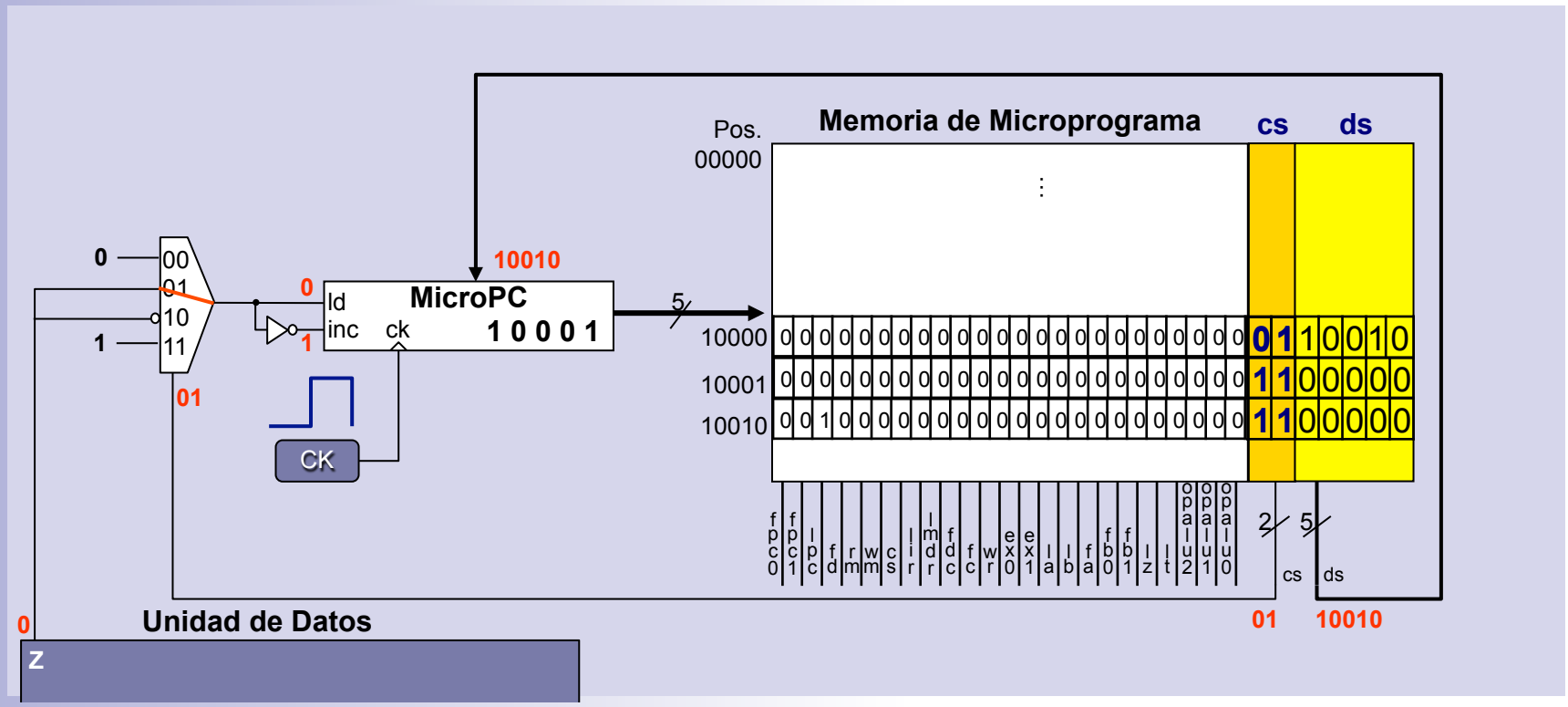
Diseño U.C. Microprogramada

- Carga condicional de MicroPC
  - Lógica para la generación de "ld" e "inc" de MicroPC
    - Entradas: "cs" selecciona condición, "Z" y "no Z"
    - Utilizamos un multiplexor para seleccionar condición salto



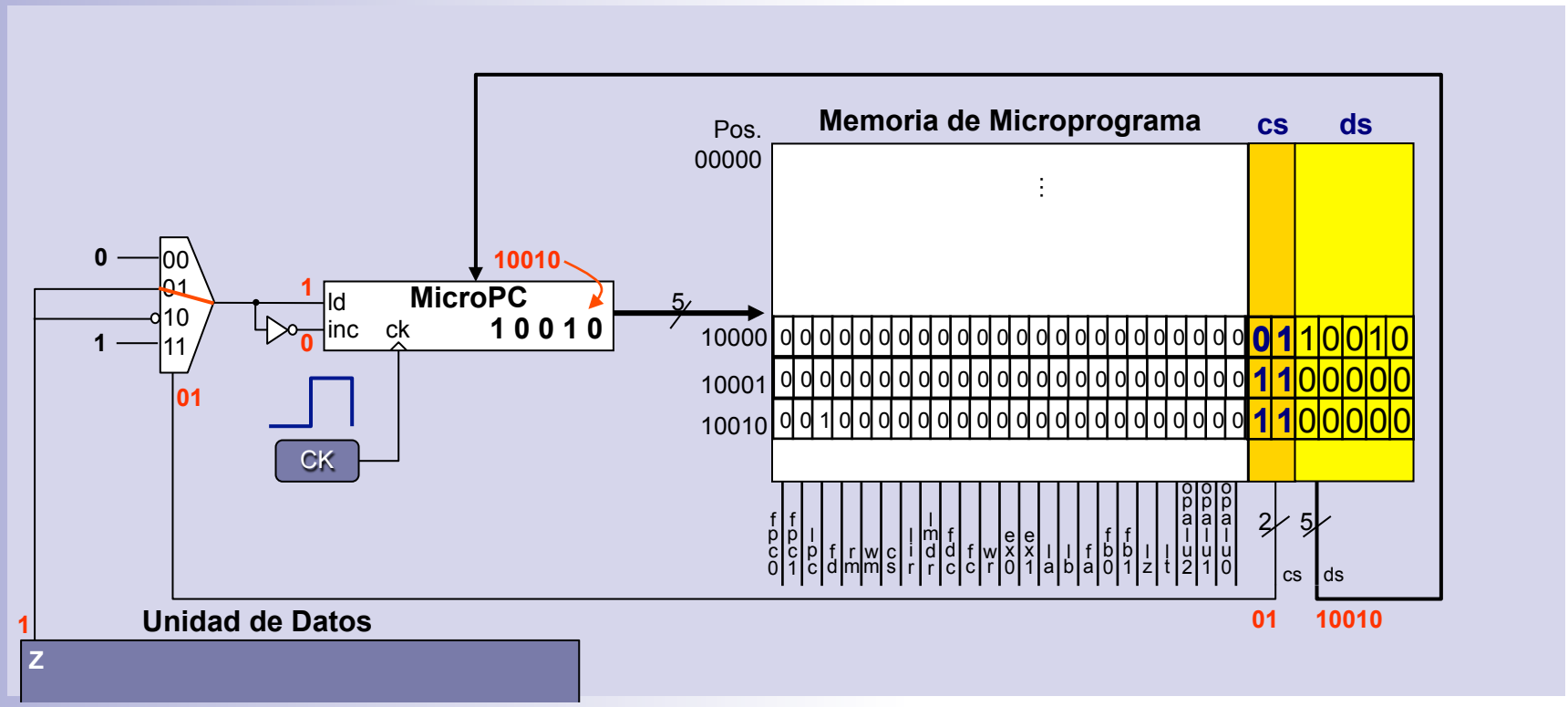
Diseño U.C. Microprogramada

- Ejecución de BEQ
  - Microinstrucción 15 (01111) salto condicional “si Z”
    - Campo cs = 01: ld = Z = 0, inc = $\overline{Z}$ = 1



Diseño U.C. Microprogramada

- Ejecución de BEQ
  - Microinstrucción 15 (01111) salto condicional “si Z”
    - Campo cs = 01: ld = Z = 1, inc = $\overline{Z}$ = 0

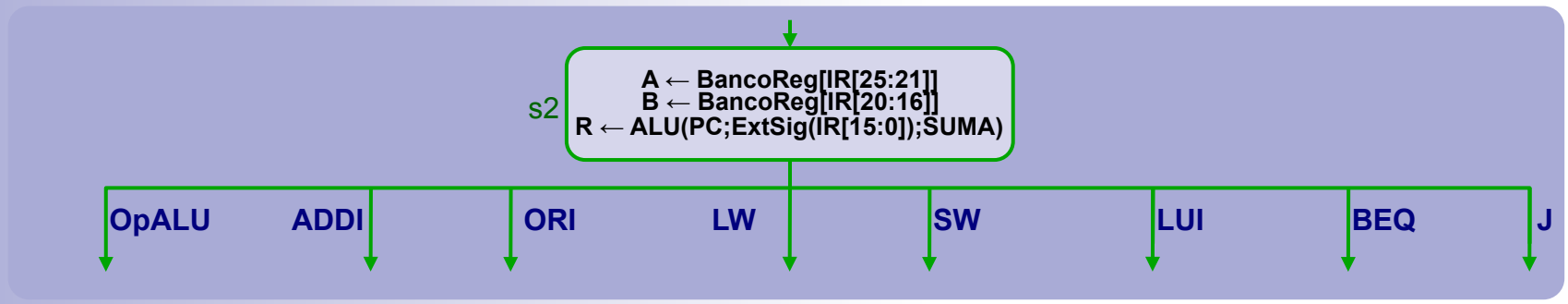


MICROPROGRAMACIÓN DEL CONJUNTO DE INSTRUCCIONES

UNIDAD DE CONTROL MICROPROGRAMADO DEL PROCESADOR R-MIPS MULTICICLO

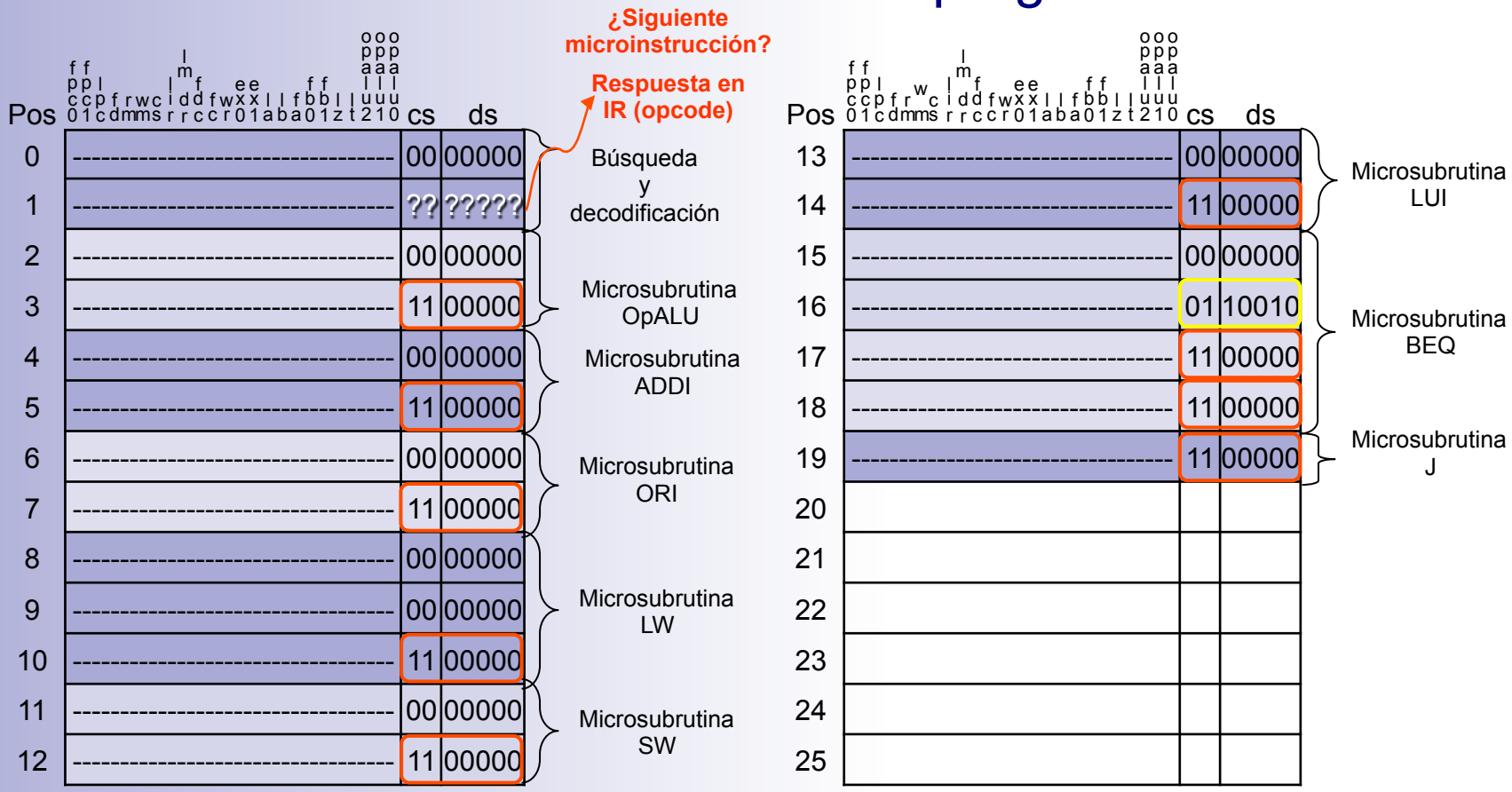
Diseño U.C. Microprogramada

- ¿Qué microinstrucción ejecutar después de la última de la fase de búsqueda?
  - Depende de la instrucción que se está ejecutando
  - Saltar a la primera microinstrucción de la microsubrutina de la instrucción en curso
    - Microsubrutina: Conjunto de microinstrucciones asociadas a la ejecución de cada instrucción máquina
  - Muchas posibles direcciones de salto



Diseño U.C. Microprogramada

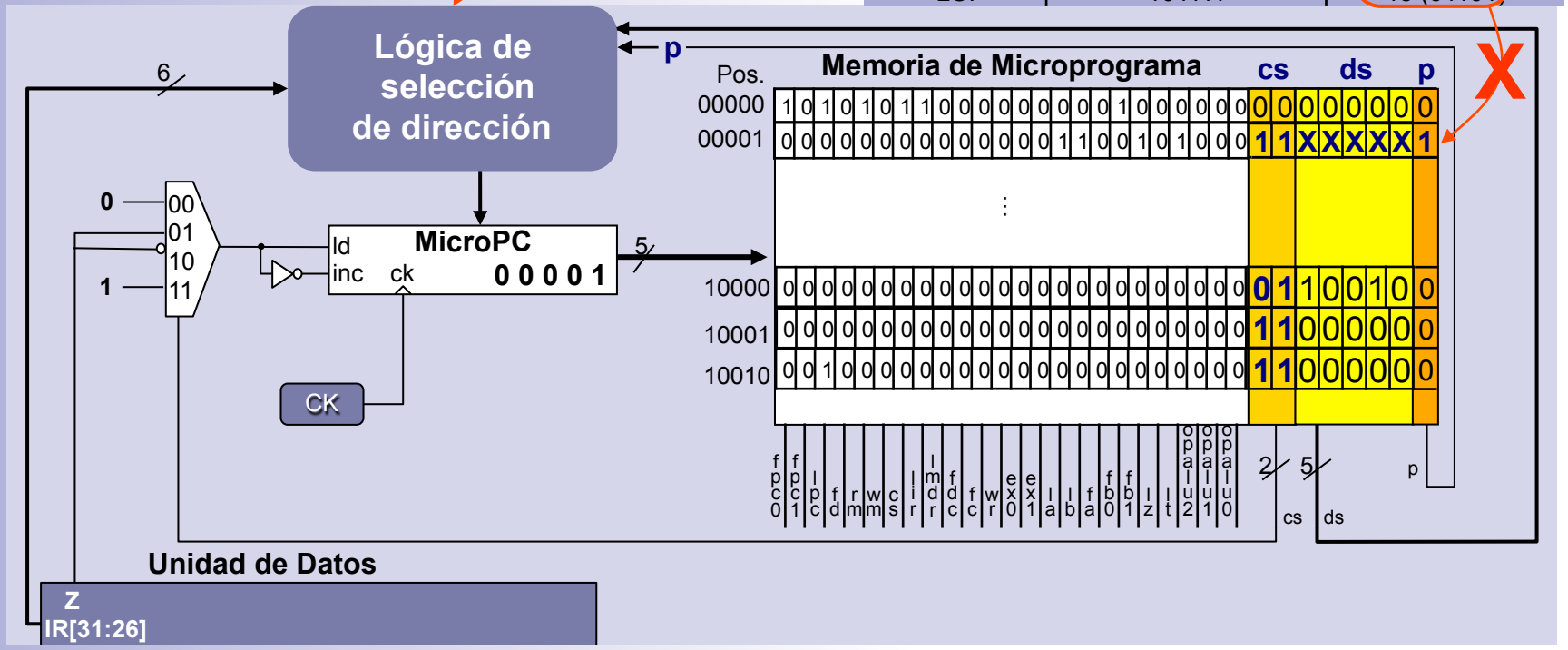
- Contenido de la memoria de microprograma



Diseño U.C. Microprogramada

- ¿A que microinstrucción saltar después de la búsqueda y decodificación?
- Crear lógica de selección de dirección
  - Decidirá valor a cargar en MicroPC
  - Se basa en "ds", "IR[31:26]" y direcciones de las primeras microinst. de cada instrucción
  - Añadir nuevo bit "p" para tomar dirección indicada por "ds" o deducible de "IR[31:26]"

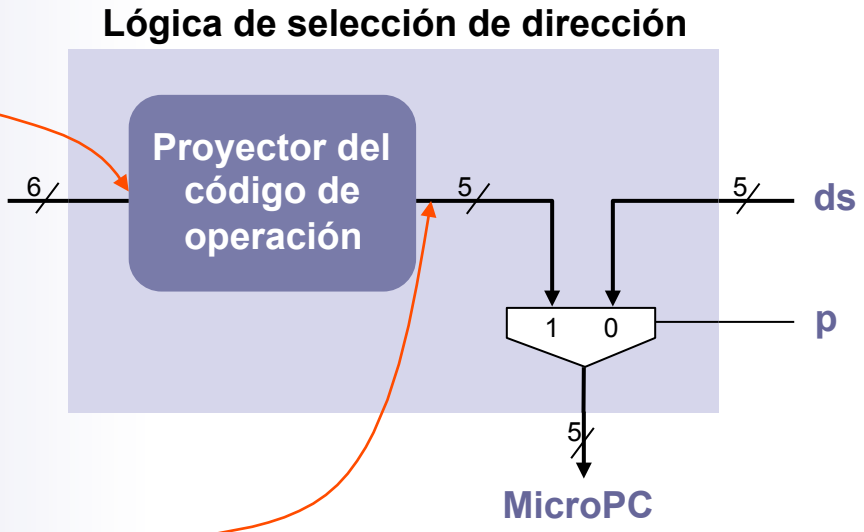
Instrucción	opcode (IR[31:26])	microinstrucción
OpALU	000000	2 (00010)
J	000010	19 (10011)
BEQ	000100	15 (01111)
ADDI	001000	4 (00100)
ORI	001101	6 (00101)
LW	100011	8 (01000)
SW	101011	11 (01011)
LUI	101111	13 (01101)



Diseño U.C. Microprogramada

- Lógica de selección de dirección
  - Determina la dirección a cargar en MicroPC
  - No determina si se carga o no
    - Sigue determinado por bits “cs” (condición de salto)
  - “p” = 0 → selecciona bits “ds” (dirección de salto)
  - “p” = 1 → genera dirección en base a IR[31:26] (Proyección del código de operación)
    - Sólo “p” = 1 en la última microinstrucción de la fase de búsqueda

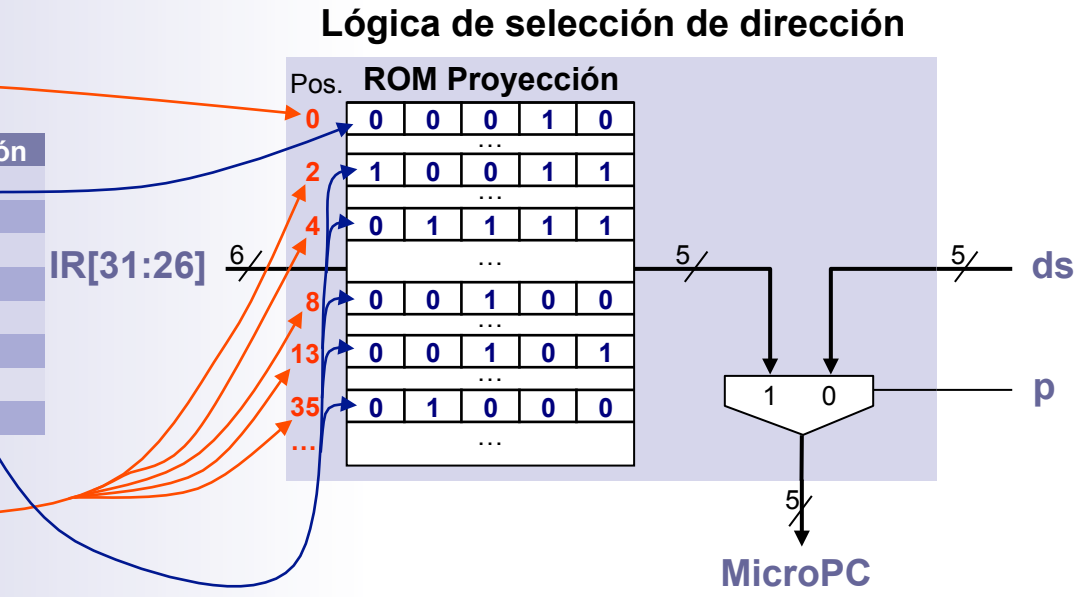
Instrucción	opcode (IR[31:26])	microinstrucción
OpALU	000000	2 (00010)
J	000010	19 (10011)
BEQ	000100	15 (01111)
ADDI	001000	4 (00100)
ORI	001101	6 (00101)
LW	100011	8 (01000)
SW	101011	11 (01011)
LUI	101111	13 (01101)



Diseño U.C. Microprogramada

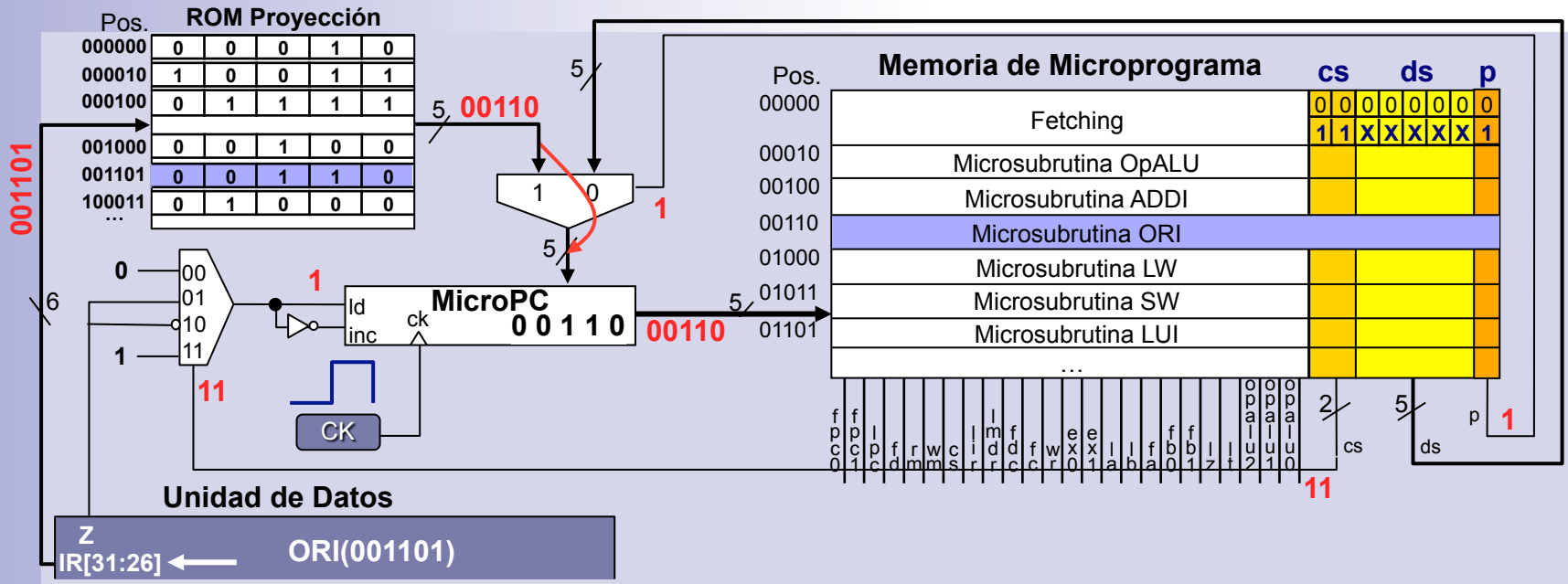
- Proyector del código de operación
  - Circuito combinacional
    - 6 bits entrada (IR[15:10])
    - 5 bits salida (Dirección de memoria de microprograma)
  - Implementación mediante ROM (ROM Proyección)
    - ROM 2^6 x 5 bits

Instrucción	opcode (IR[31:26])	microinstrucción
OpALU	000000	2 (00010)
J	000010	19 (10011)
BEQ	000100	15 (01111)
ADDI	001000	4 (00100)
ORI	001101	6 (00101)
LW	100011	8 (01000)
SW	101011	11 (01011)
LUI	101111	13 (01101)



Diseño U.C. Microprogramada

- Proyección del código de operación
  - Microinstrucción 2 → "p" = 1, "cs" = 11
  - Siguiente microinstrucción viene de la ROM de Proyección
  - Depende del código de operación (opcode, IR[31:26]) de la instrucción cargada en IR (microinstrucción 0)
  - No se puede proyectar en la misma microinstrucción que carga IR



Diseño U.C. Microprogramada

- Proyección del código de operación
  - Microinstrucción 2 → "p" = 1, "cs" = 11
  - Siguiente microinstrucción viene de la ROM de Proyección
  - Depende del código de operación (opcode, IR[31:26]) de la instrucción cargada en IR (microinstrucción 0)
  - No se puede proyectar en la misma microinstrucción que carga IR