La unidad de control

Planteamiento

- **Aspectos a estudiar:**

  - **Procesamiento**
    - Fases de ejecución de las instrucciones
    - Operaciones elementales
    - Cronogramas de ejecución de las instrucciones

  - **Diseño de la unidad de control**
    - Control cableado
    - Control microprogramado

La unidad de control

Bibliografía

- ***Organización y arquitectura de computadores*. William Stallings. Pearson-Prentice Hall (7ª edición)**
  - ➔ **Capítulo 3 desde el comienzo hasta el apartado *Interrupciones***
  - ➔ **Capítulo 16 completo y 17 hasta el apartado *Formato de microinstrucción del LSI-11***

- ***Arquitectura de computadores. Un enfoque cuantitativo*. John L. Hennessy y David A. Patterson. Mc Graw Hill, 1993**
  - ➔ **Capítulo 5 hasta el apartado *Interrupciones***

La unidad de control

Índice

La unidad de control

1. Introducción

- La ejecución de un programa implica la ejecución de la secuencia de instrucciones que lo componen
- La ejecución de cada instrucción implica la ejecución de una secuencia de fases sencillas
- Cada fase de ejecución de una instrucción se descompone en una serie de operaciones elementales
- La unidad de control rige este proceso y genera las señales de control que gobiernan la ruta de datos

La unidad de control

1. Introducción

- La unidad de control gobierna el proceso enumerado en la diapositiva anterior
- Su función está directamente relacionada con el flujo de instrucciones
- Depende fuertemente de las características del repertorio de instrucciones
- Consume aproximadamente el 50% de los transistores del procesador pero su diseño es mucho más complejo que el de la ruta de datos

La unidad de control

1. Introducción

La unidad de control

2. Fases de ejecución de una instrucción

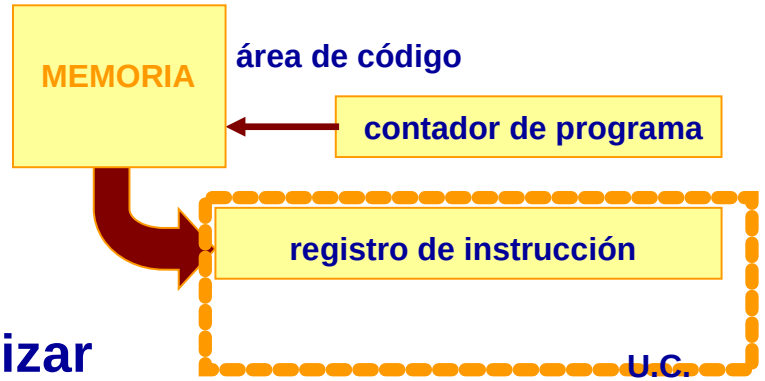
- **Las fases de ejecución de una instrucción son:**

  - ➔Búsqueda (*fetch*)
  - ➔Decodificación
  - ➔Lectura de operandos
  - ➔Operación
  - ➔Escritura de resultados

La unidad de control

2. Fases de ejecución de una instrucción

- **Búsqueda (*fetch*)**
  - ➔ **Traer de memoria la instrucción a ejecutar**
    - ➔ apuntada por el CP
    - ➔ al registro de instrucción



- **Decodificación**
  - ➔ **Interpretar la operación a realizar**

  - ➔ **Entre estas dos fases hay que <u>actualizar el CP</u>**
    - ➔ Formato de tamaño regular
    - ➔ Formato de tamaño variable

- **El tiempo de decodificación de la instrucción influye en la velocidad de reloj**

  ➔ **Queremos que la decodificación se realice en un único ciclo de reloj**

    ➔ Un formato de instrucciones complejo dificulta la decodificación y por tanto puede alargar el periodo de reloj

La unidad de control

2. Fases de ejecución de una instrucción

- **Lectura de operandos**
  - ➔ **Buscar los operandos en función de los modos de direccionamiento**
    - ➔ Esta fase puede estar integrada en la decodificación
- **Operación**
  - ➔ **Realizar la operación**
- **Escritura de resultados**
  - ➔ **Escribir el resultado en la ubicación determinada por los modos de direccionamiento**

La unidad de control

2. Fases de ejecución de una instrucción

- **Estas fases son válidas aunque la operación a realizar no sea de proceso**

  - ➔ **Si es una transferencia, la fase de ejecución se reduce a la copia de un dato desde el operando fuente al destino**

  - ➔ **Si es una bifurcación, la evaluación de la condición se suele llevar a cabo durante la fase de decodificación**
    - ➔ Si el salto se toma, se actualiza el contador de programa en las fases siguientes
    - ➔ Si el salto no se toma, las fases siguientes no hacen nada

La unidad de control

2.1. Operaciones elementales

- **Cada una de las fases de ejecución de una instrucción se descompone en una serie de operaciones más sencillas conocidas como OPERACIONES ELEMENTALES**

- **Dos tipos:**

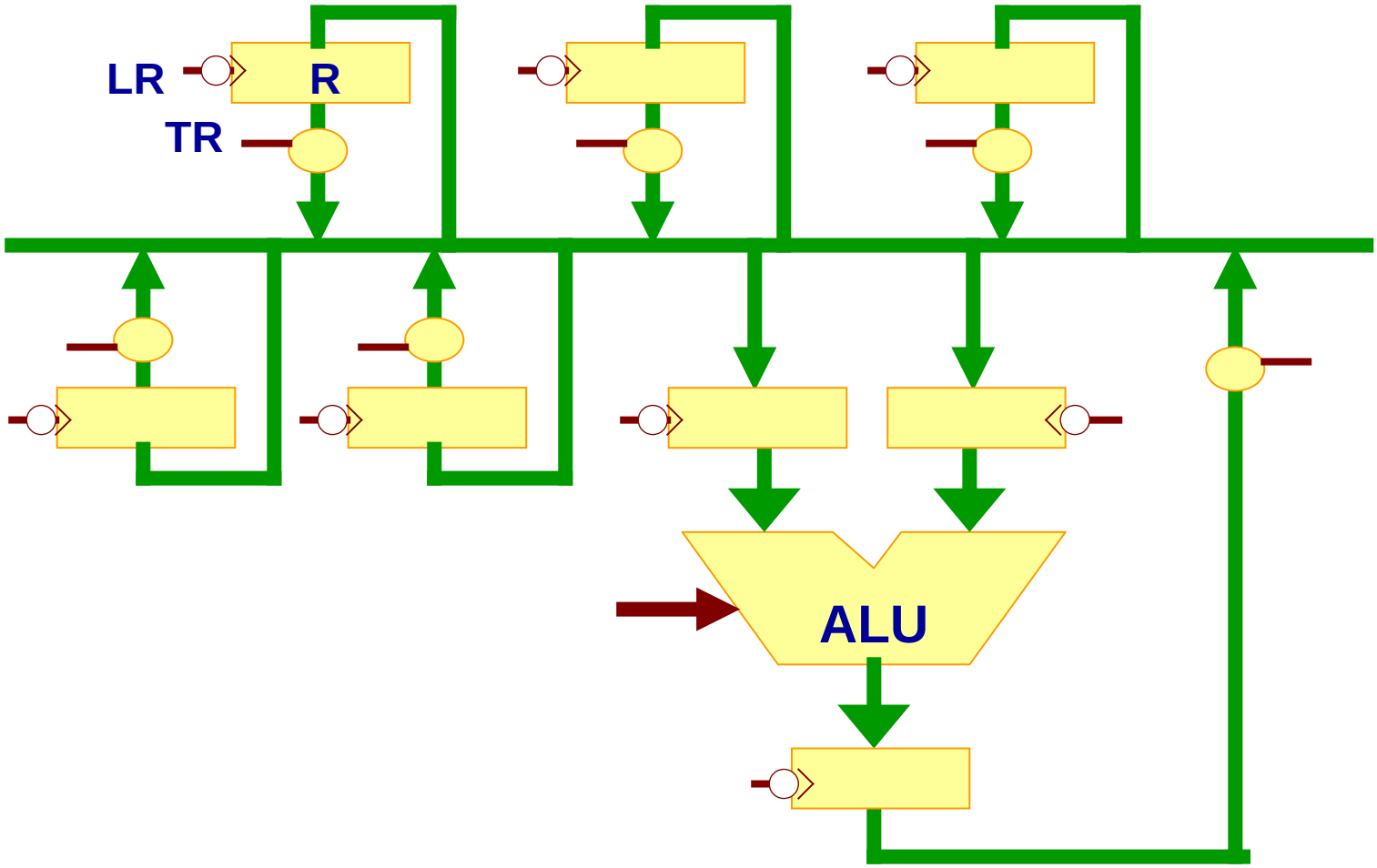
  ➔ **Operaciones elementales de transferencia**

  ➔ **Operaciones elementales de proceso**

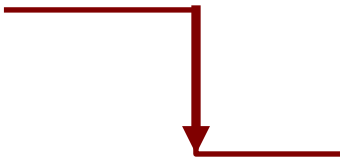
La unidad de control

2.1. Operaciones elementales

2.1.1. Las señales de control



- **Descripción de los elementos *hardware***
  - ➔ **Registros:**
    - ➔ Elementos de almacenamiento
    - ➔ Dispositivos síncronos
    - ➔ Activos por flanco
      - De bajada (para nosotros)



  - ➔ ***Buffers* triestado:**
    - ➔ Separan los elementos de almacenamiento del bus
    - ➔ Impiden que el bus sea cargado o que varios elementos vuelquen datos al mismo tiempo
    - ➔ Activos por nivel

La unidad de control

2.1. Operaciones elementales

2.1.1. Las señales de control

- **Advertencias**
  - **Nombrar las señales con alguna regla de construcción**
    - L*xx* -> cargas
    - T*xx* -> transferencias
  - **Los elementos de almacenamiento tienen dos conexiones al bus**
    - Una para escribir (a través del *buffer*)
    - Otra para leer (directa ya que es la U.C. quien determina el instante y el elemento que se carga)
  - **LOS BUSES NO ALMACENAN INFORMACIÓN**

La unidad de control

2.1. Operaciones elementales

2.1.2. De transferencia

- **Las operaciones elementales de transferencia se encargan de mover datos entre registros**

- **Las señales de control involucradas realizan 2 funciones:**
  - ➔ **Establecer un camino entre el origen y el destino**
  - ➔ **Copiar un dato en un registro**

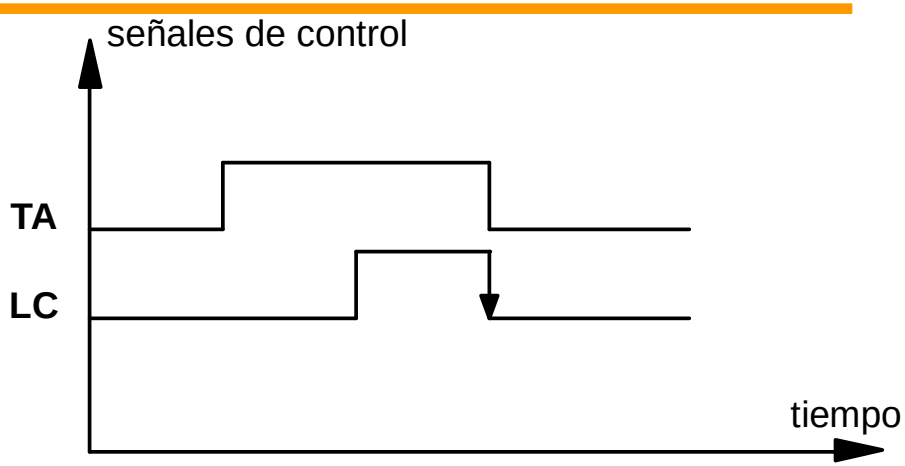
La unidad de control

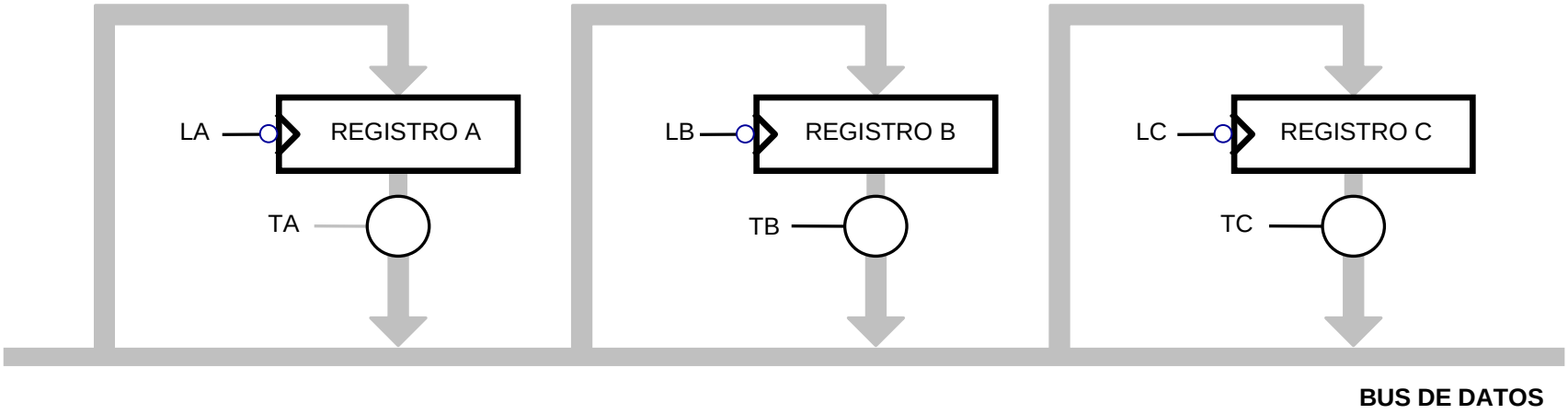
2.1. Operaciones elementales

2.1.2. De transferencia

- **Transferencia entre registros**
  - ➔ **Establecer camino**
  - ➔ **Salvar en registro**

- **Ejemplo:**





La unidad de control

2.1. Operaciones elementales

2.1.3. De proceso

- **Las operaciones elementales de proceso se encargan de procesar datos en un operador**

- **Las señales de control involucradas realizan 2 funciones:**
  - ➔ **Proporcionar operandos a un operador**
    - ➔ Establecer un camino entre los operandos y las entradas del operador
  - ➔ **Salvar el resultado en un registro**

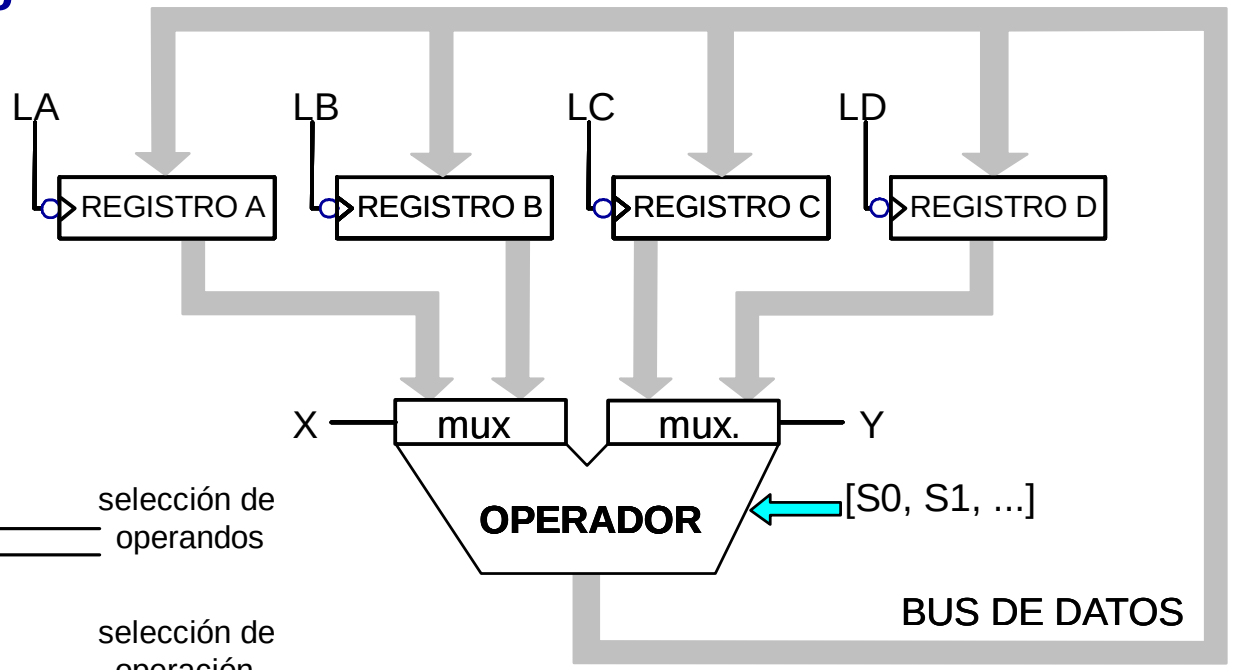
La unidad de control

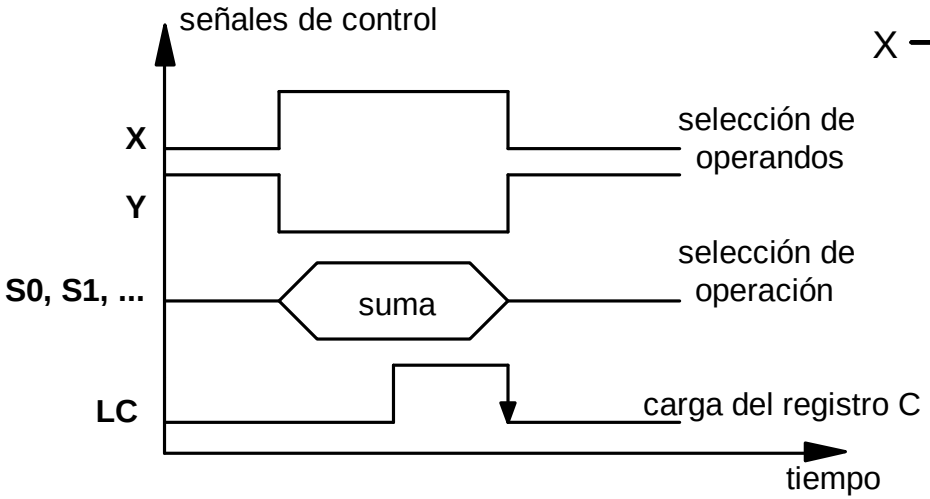
2.1. Operaciones elementales

2.1.3. De proceso

- **Proceso**
  - ➔ **Establecer camino a través de un operador**
  - ➔ **Salvar en registro**

- **Ejemplo:**





La unidad de control

2.1. Operaciones elementales

2.1.3. De proceso

- **El tiempo de retardo del operador más común influye en la velocidad de reloj**

  ➔ **Normalmente el operador de suma**

    ➔ No es el más rápido pero sí el más frecuentemente usado

La unidad de control

2.2. Cronogramas

- La implementación de cada instrucción del repertorio supone realizar una secuencia de operaciones elementales en las que se activan las señales de control precisas para llevarla a cabo

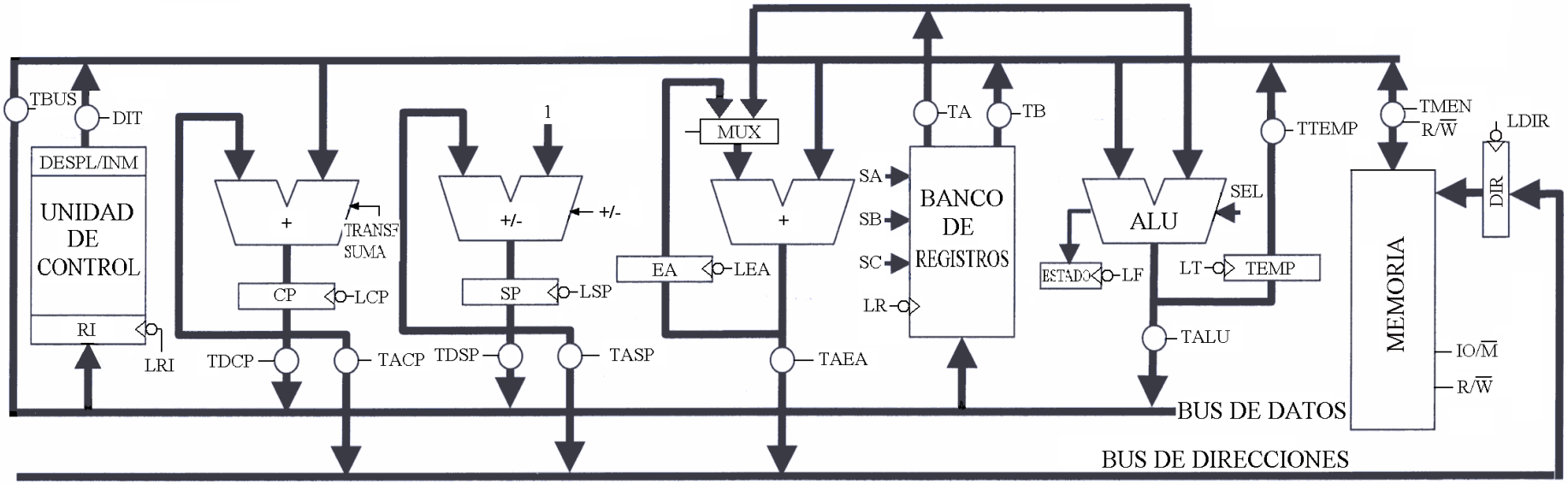
- La secuencia completa se conoce como cronograma de ejecución

- A continuación se propone una máquina sobre la que podemos generar cronogramas

La unidad de control

2.2. Cronogramas

- **Sea la siguiente ruta de datos:**



- Palabra: 16 bits
- Bus datos y direcciones: 16 bits
- Memoria organizada en palabras

- **Características de la arquitectura propuesta:**
  - ➔ **Multiplicidad funcional (paralelismo posible)**
  - ➔ **Admite instrucciones de tamaño variable**
    - ➔ Unidad de actualización del CP conectada a la UC
  - ➔ **No es posible trabajar con dos datos de memoria por falta de buses de datos**
    - ➔ ¡OJO! Si elimino TBUS se pueden provocar conflictos sobre el bus

- **Fase de búsqueda (descomposición en op. elementales)**

ACCIÓN	TIPO
• Transferir CP a DIR	op. elemental de transferencia

  - ➔ Establecer camino → TACP
  - ➔ Salvar información → LDIR

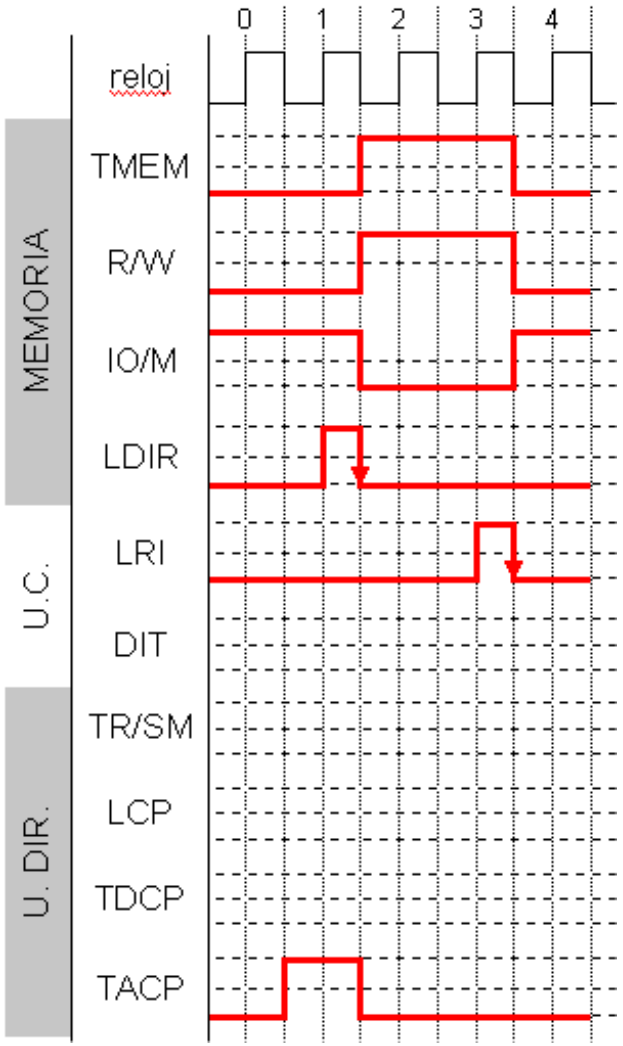
• Transferir contenido de memoria a RI	op. elemental de transferencia

  - ➔ Establecer camino → TMEM, R/W, IO/M
  - ➔ Salvar información → LRI

La unidad de control

2.2. Cronogramas

- **Cronograma de la fase de búsqueda**



- **Actualización del CP (descomposición en op. elementales)**

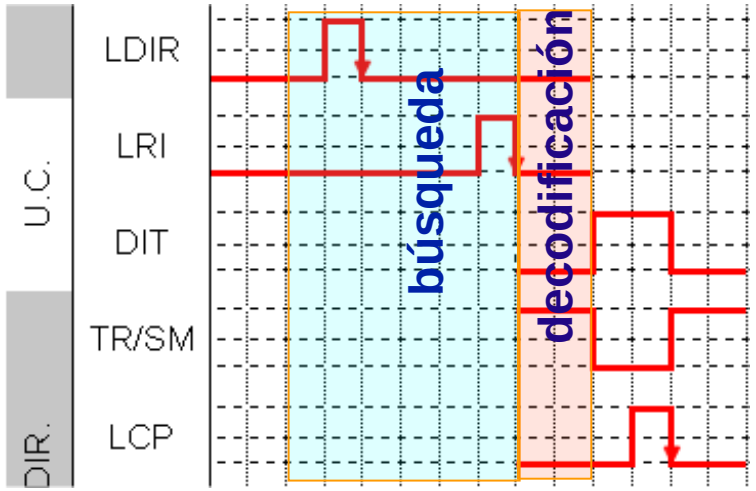
ACCIÓN	TIPO
• Sumar CP con longitud	op. elemental de proceso

  - ➔ Transferir longitud desde UC → DIT
  - ➔ Operación de suma → TR/SM
  - ➔ Salvar nuevo CP → LCP

La unidad de control

2.2. Cronogramas

- **Cronograma de la actualización del PC**



- **Operación de suma** (descomposición en op. elementales)

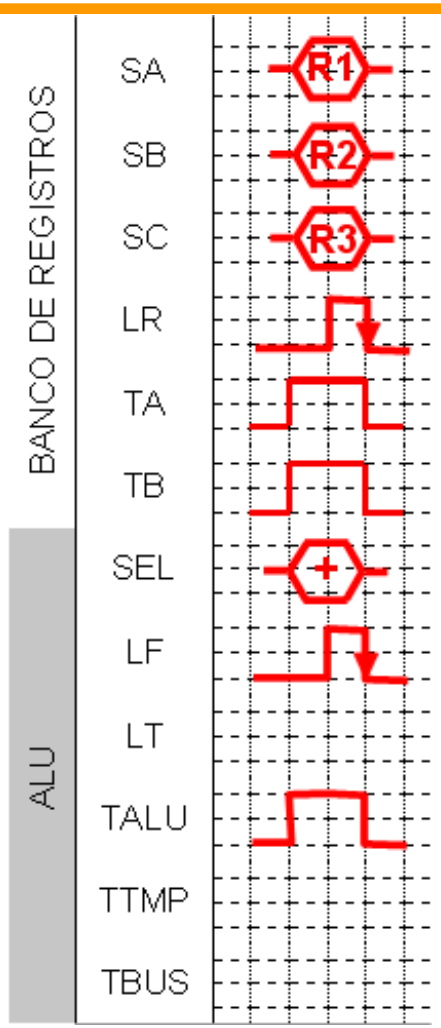
ACCIÓN	TIPO
• Sumar R1 + R2 → R3	op. elemental de proceso

  - → Seleccionar R1, R2, R3 → SA, SB, SC
  - → Establecer camino con la ALU → TA, TB
  - → Seleccionar operación de suma → SEL
  - → Establecer camino con el banco de registros → TALU
  - → Salvar resultado y estado → LR, LF

La unidad de control

2.2. Cronogramas

- **Cronograma de una operación de proceso entre registros**
  - ➔ **Es posible realizar todo el proceso en un sólo ciclo de reloj**



La unidad de control

La unidad de control

3. El diseño de la UC

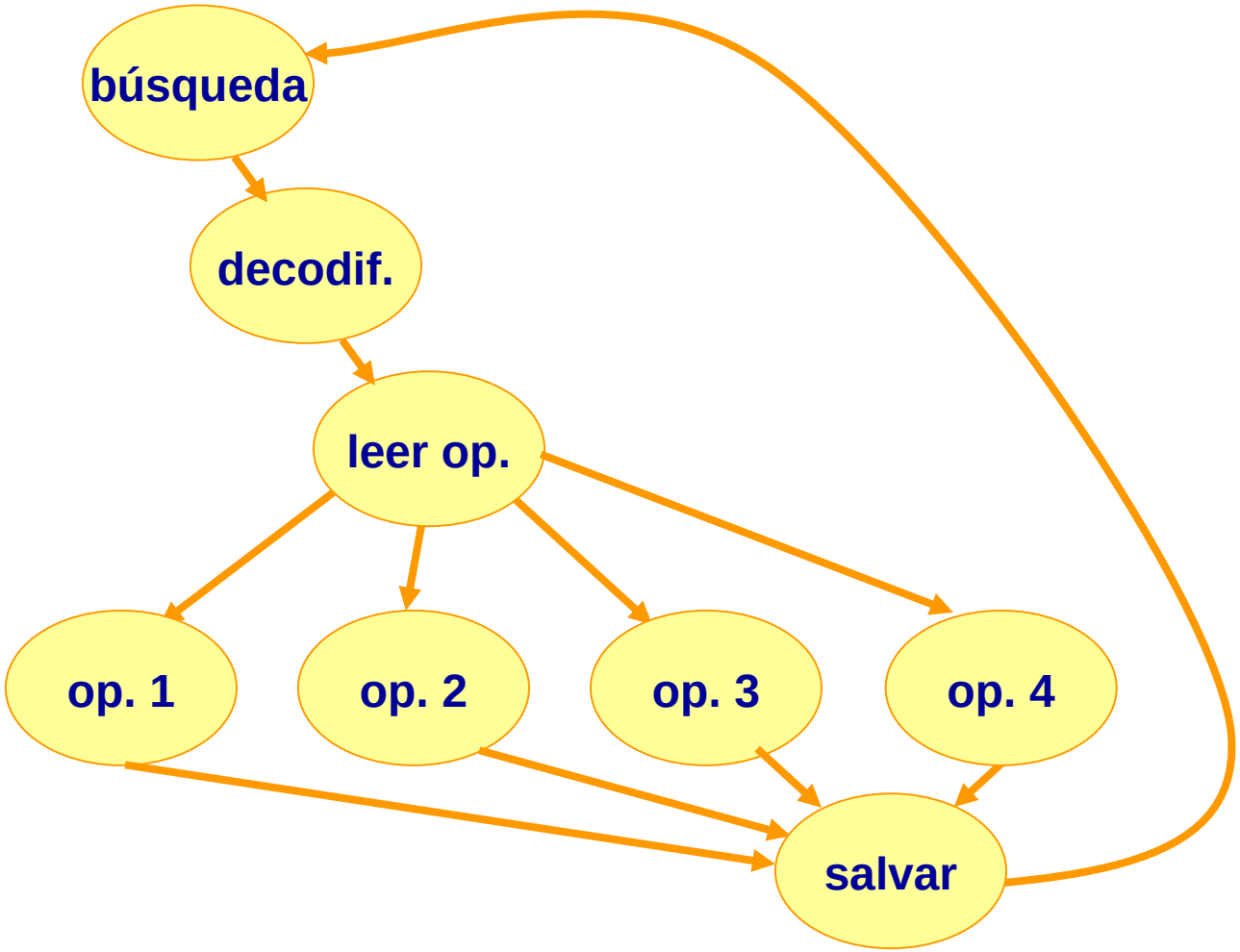
- **Diseño de un computador**
  - ➔ **Ruta de datos**
    - ➔ Objetivos de coste y rendimiento
    - ➔ Impone periodo de reloj
    - ➔ 50% de los transistores
    - ➔ Diseño de operadores ➔ **muy simple**
    - ➔ Diseño del banco de registros ➔ **sencillo**
  - ➔ **Unidad de control**
    - ➔ **Verdaderamente difícil**
    - ➔ 5 fases de ejecución (búsqueda, decodificación, leer operandos, ejecutar, escribir resultado)
    - ➔ Multitud de operaciones y modos de direccionamiento

La unidad de control

3. El diseño de la UC

- **El diseño de la UC plantea varios retos:**
  - **En primer lugar, el diseño propiamente dicho**
    - Correcto
    - Rápido
  - **El coste (en términos de área de silicio)**
  - **El procedimiento de verificación del correcto funcionamiento**
    - Que sea fácilmente depurable
    - Su capacidad de adaptación y compatibilidad binaria

- **Ejemplo: máquina de estados sencilla con 4 operaciones**



- **Diseño del circuito secuencial**
  - ➔ 8 estados → 3 bits
  - ➔ 4 operaciones → 2 bits

estado actual	estado siguiente			
	00	01	10	11
000	0 0 1			
001	0 1 0			
010	0 1 1	1 0 0	1 0 1	1 1 0
011	1 1 1			
100	1 1 1			
101	1 1 1			
110	1 1 1			
111	0 0 0			

- **Bit de menor peso del estado siguiente:**

$$D_0 = \bar{Q}_2 \cdot \bar{Q}_1 \cdot \bar{Q}_0 + \bar{Q}_2 \cdot Q_1 \cdot Q_0 + Q_2 \cdot \bar{Q}_0 + Q_2 \cdot \bar{Q}_1 + \bar{Q}_2 \cdot Q_1 \cdot \bar{Y}$$

D_0		estados							
		$Q_2 = 0$				$Q_2 = 1$			
	Q_1Q_0 / XY	00	01	11	10	00	01	11	10
control	00	1	0	1	1	1	1	0	1
	01	1	0	1	0	1	1	0	1
	11	1	0	1	0	1	1	0	1
	10	1	0	1	1	1	1	0	1

- **Bit intermedio del estado siguiente:**

$$D_1 = Q_2 \cdot \bar{Q}_0 + \bar{Q}_2 \cdot Q_0 + Q_2 \cdot \bar{Q}_1 + \bar{Q}_2 \cdot Q_1 \cdot \bar{X} \cdot \bar{Y} + \bar{Q}_2 \cdot Q_1 \cdot X \cdot Y$$

D_1		estados							
		$Q_2 = 0$				$Q_2 = 1$			
Q_1Q_0 / XY		00	01	11	10	00	01	11	10
control	00	0	1	1	1	1	1	0	1
	01	0	1	1	0	1	1	0	1
	11	0	1	1	1	1	1	0	1
	10	0	1	1	0	1	1	0	1

- **Bit de mayor peso del estado siguiente:**

$$D_2 = \bar{Q}_2 \cdot Q_1 \cdot Q_0 + Q_2 \cdot \bar{Q}_0 + Q_2 \cdot \bar{Q}_1 + \bar{Q}_2 \cdot Q_1 \cdot Y + \bar{Q}_2 \cdot Q_1 \cdot X$$

		estados							
	D_2	$Q_2 = 0$				$Q_2 = 1$			
	Q_1Q_0 / XY	00	01	11	10	00	01	11	10
control	00	0	0	1	0	1	1	0	1
	01	0	0	1	1	1	1	0	1
	11	0	0	1	1	1	1	0	1
	10	0	0	1	1	1	1	0	1

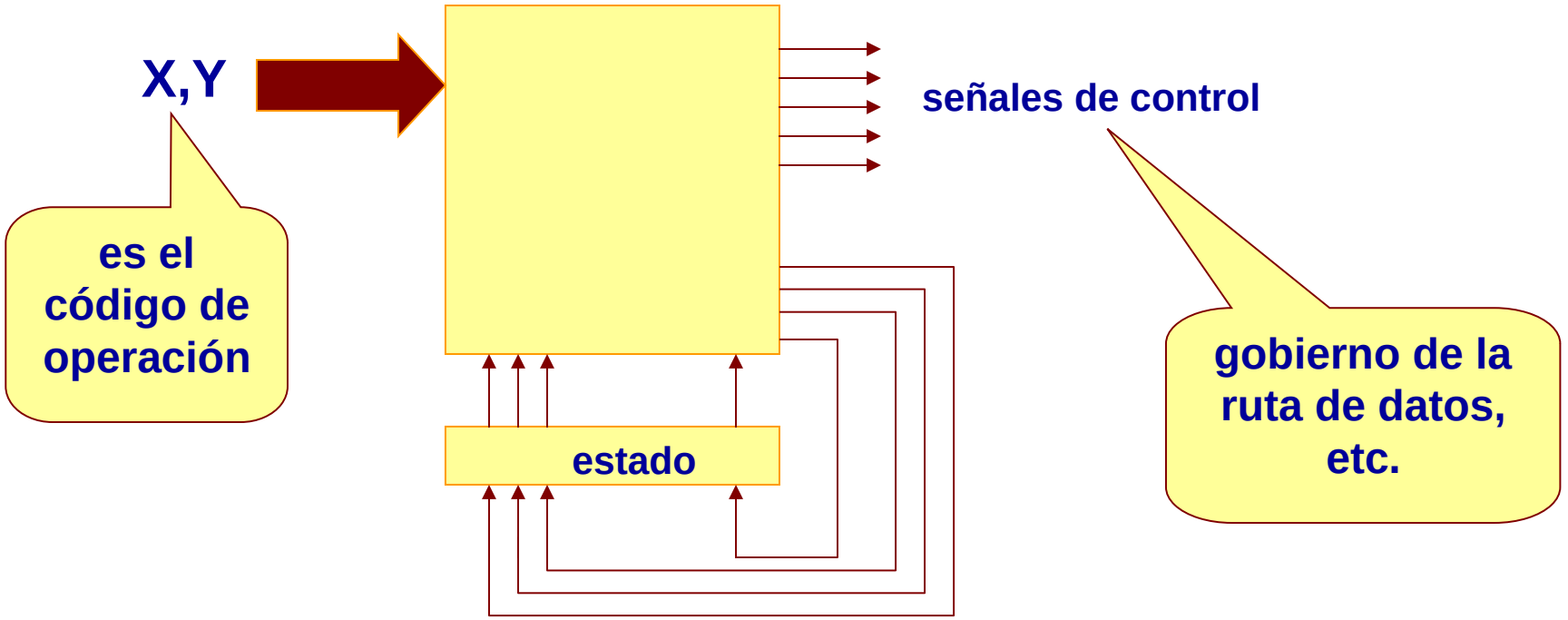
La unidad de control

3. El diseño de la UC

$$D_0 = \bar{Q}_2 \cdot \bar{Q}_1 \cdot \bar{Q}_0 + \bar{Q}_2 \cdot Q_1 \cdot Q_0 + Q_2 \cdot \bar{Q}_0 + Q_2 \cdot \bar{Q}_1 + \bar{Q}_2 \cdot Q_1 \cdot \bar{Y}$$

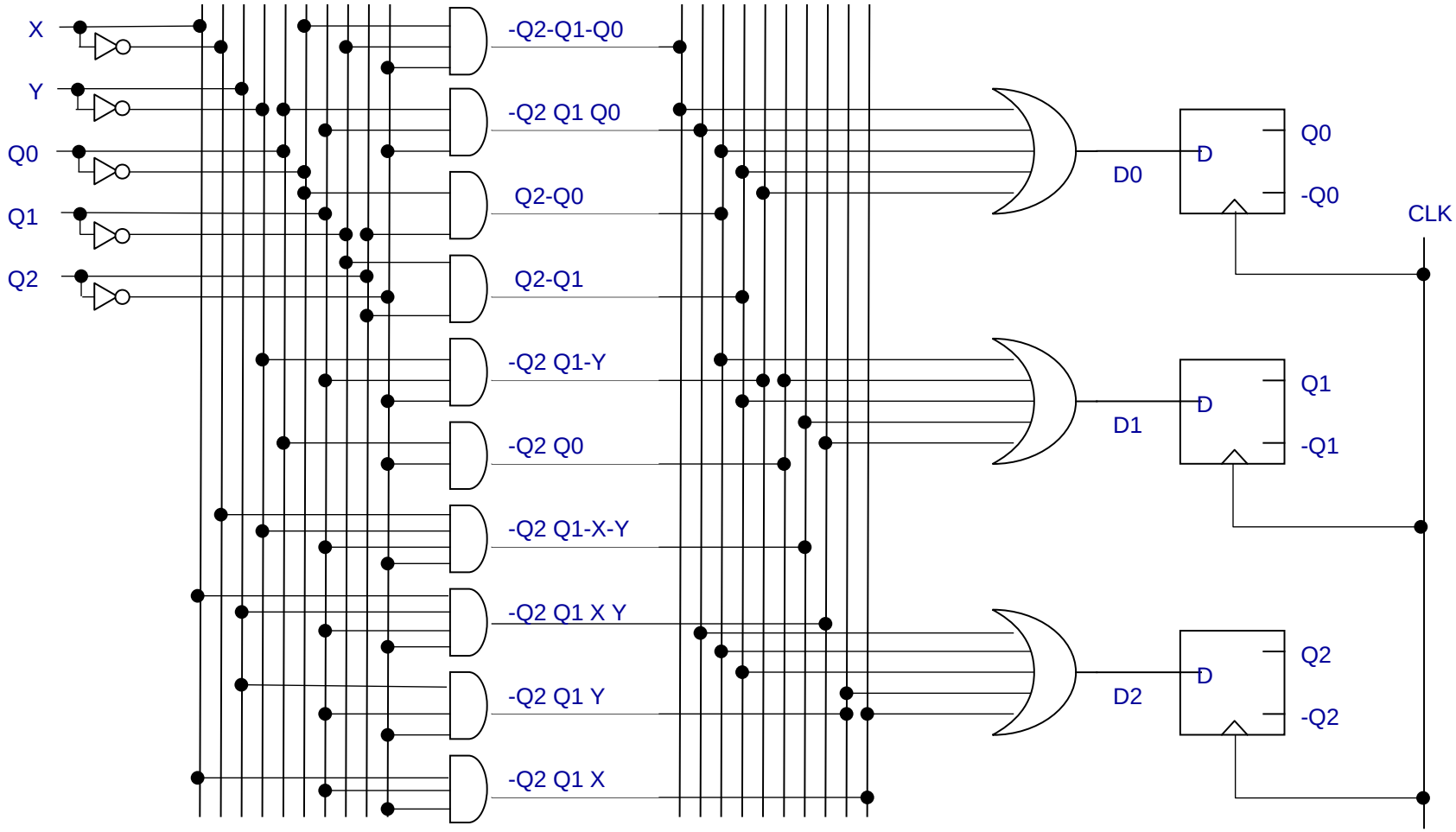
$$D_1 = Q_2 \cdot \bar{Q}_0 + \bar{Q}_2 \cdot Q_0 + Q_2 \cdot \bar{Q}_1 + \bar{Q}_2 \cdot Q_1 \cdot \bar{X} \cdot \bar{Y} + \bar{Q}_2 \cdot Q_1 \cdot X \cdot Y$$

$$D_2 = \bar{Q}_2 \cdot Q_1 \cdot Q_0 + Q_2 \cdot \bar{Q}_0 + Q_2 \cdot \bar{Q}_1 + \bar{Q}_2 \cdot Q_1 \cdot Y + \bar{Q}_2 \cdot Q_1 \cdot X$$



La unidad de control

3. El diseño de la UC

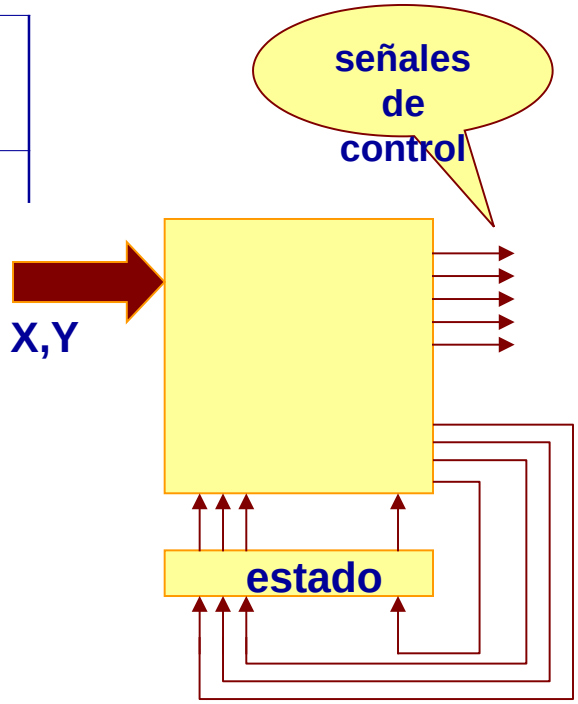


La unidad de control

3. El diseño de la UC

➔ **Hay que completar el circuito secuencial con la red combinacional que asigna valores a las señales de control en cada estado**

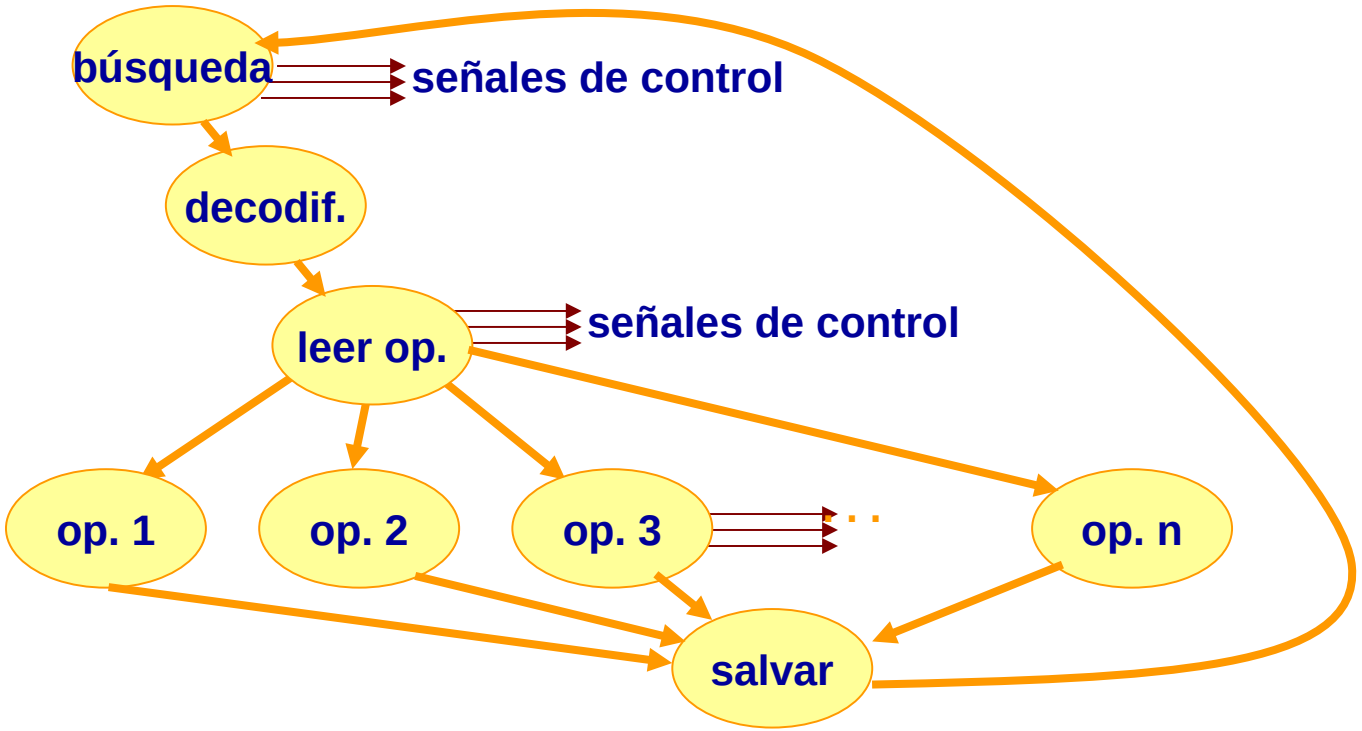
estado actual	señales de control							
	. . .	. . .	TACP	LDIR	TMEM	R/W	LRI	. . .
000	0	0	1	1	1	1	1	0
001								
010								
011								
100								
101								
110								
111								



La unidad de control

3. El diseño de la UC

- **En realidad no hay 4 operaciones sino cientos…**



- **Cada fase de ejecución es también una máquina de estados**
  - Las fases de lectura de operandos y de escritura de resultados tienen una máquina de estados para cada modo de direccionamiento
  - La fase de ejecución tiene una máquina de estados diferente para cada operación

¡¡ EXPLOSIÓN DE ESTADOS !!

- **¿Cuál es dimensión del problema?**
  - **Juego de 128 instrucciones ➔ 7 bits**
  - **Algunas instrucciones se desarrollan en decenas de ciclos**
  - **Gran número de señales de control:**
    - Intel 8086 → 150 señales
    - Motorola 68000 → 70 señales
    - Procesadores segmentados → cientos de señales

¡Problema bastante importante!

La unidad de control

3. El diseño de la UC

- **Dos métodos de diseño:**

  - **UC de lógica cableada**
    - Basado en el diseño de circuitos secuenciales

  - **UC de lógica almacenada o microprogramada**
    - Las secuencias de gobierno de la ruta de datos se almacenan en memoria

La unidad de control

3. El diseño de la UC

- **Lógica cableada (método tradicional de diseño lógico)**

  - **PROS:**
    - Circuito más rápido que el de lógica almacenada
    - Circuito mínimo

  - **CONTRAS:**
    - Muy laborioso de diseñar (aunque el CAD ayuda mucho)
    - Muy difícil de modificar y depurar

La unidad de control

3. El diseño de la UC

- **Lógica almacenada**

  - **PROS:**
    - Fácil de depurar ➔ acorta el tiempo de desarrollo
    - Simplifica la compatibilidad binaria

  - **CONTRAS:**
    - Es lento (lectura de memoria)
    - Ocupa mucho sitio ➔ gran número de señales de control

La unidad de control

3. El diseño de la UC

3.1. Control cableado

- **Tipos:**

  - **Circuito secuencial**
    - Diseño óptimo

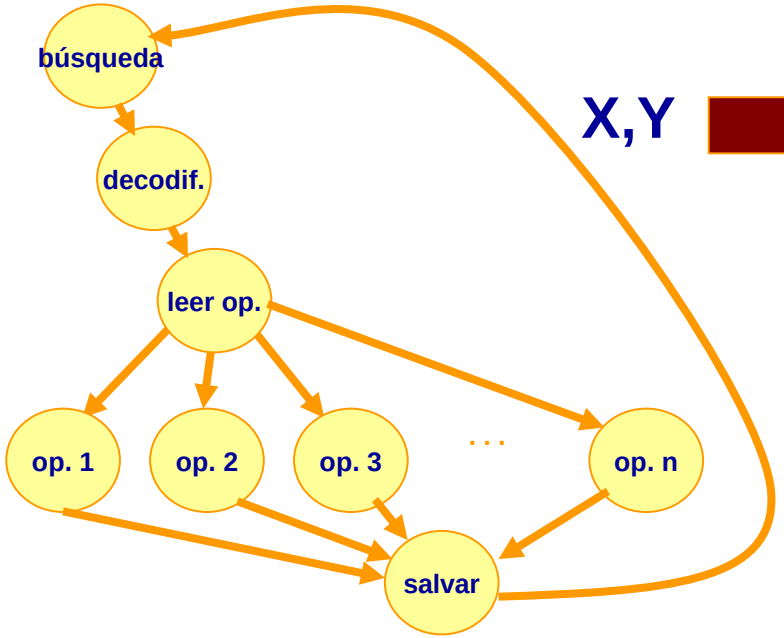
  - **Células de retardo**

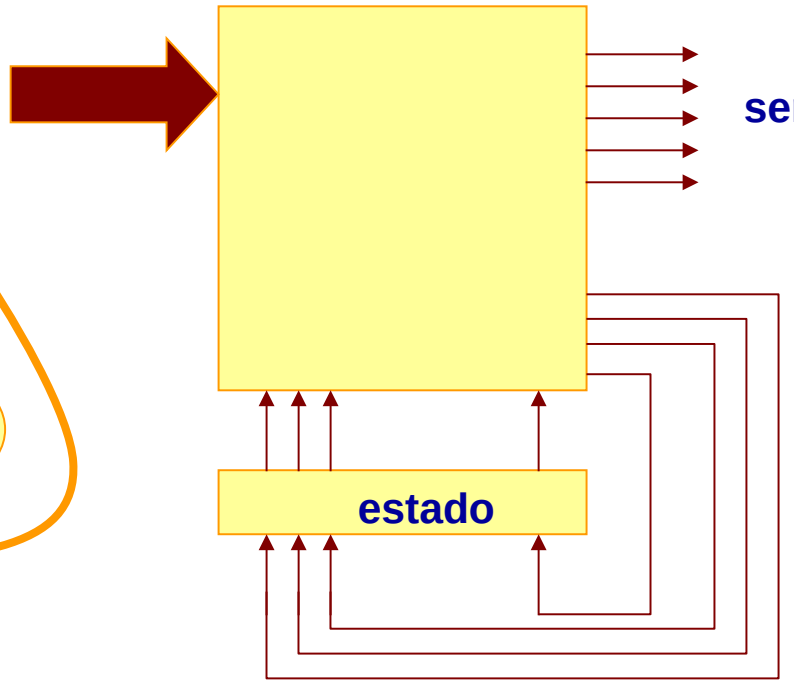
- **Circuito secuencial**

$$D_0 = \bar{Q}_2 \cdot \bar{Q}_1 \cdot \bar{Q}_0 + \bar{Q}_2 \cdot Q_1 \cdot Q_0 + Q_2 \cdot \bar{Q}_0 + Q_2 \cdot \bar{Q}_1 + \bar{Q}_2 \cdot Q_1 \cdot \bar{Y}$$

$$D_1 = Q_2 \cdot \bar{Q}_0 + \bar{Q}_2 \cdot Q_0 + Q_2 \cdot \bar{Q}_1 + \bar{Q}_2 \cdot Q_1 \cdot \bar{X} \cdot \bar{Y} + \bar{Q}_2 \cdot Q_1 \cdot X \cdot Y$$

$$D_2 = \bar{Q}_2 \cdot Q_1 \cdot Q_0 + Q_2 \cdot \bar{Q}_0 + Q_2 \cdot \bar{Q}_1 + \bar{Q}_2 \cdot Q_1 \cdot Y + \bar{Q}_2 \cdot Q_1 \cdot X$$



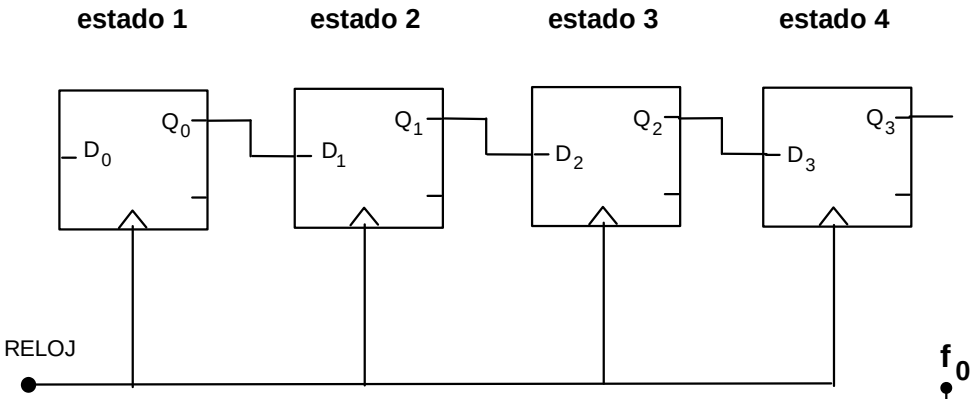


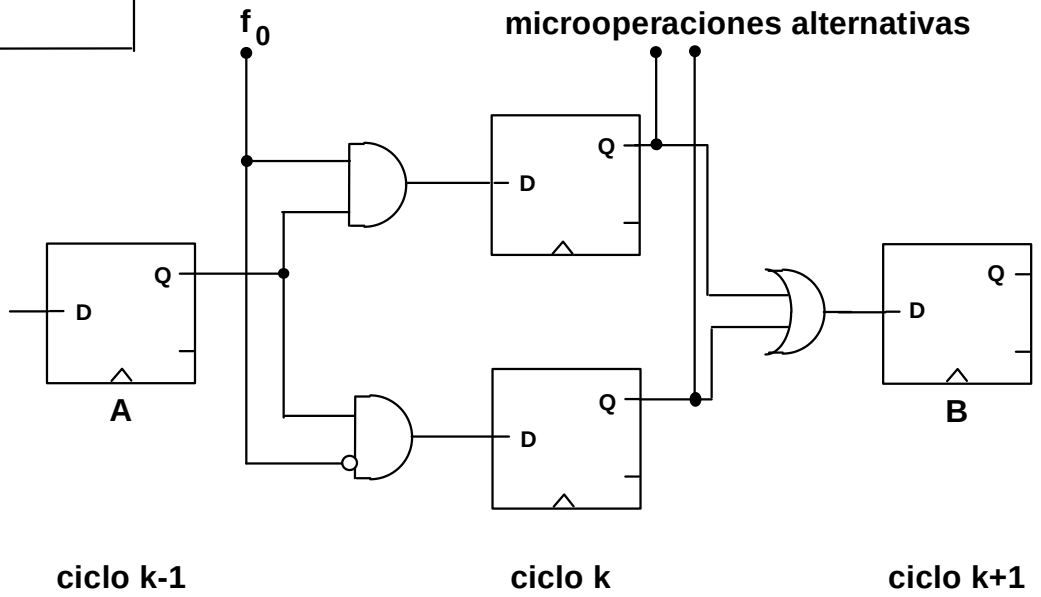
La unidad de control

3. El diseño de la UC

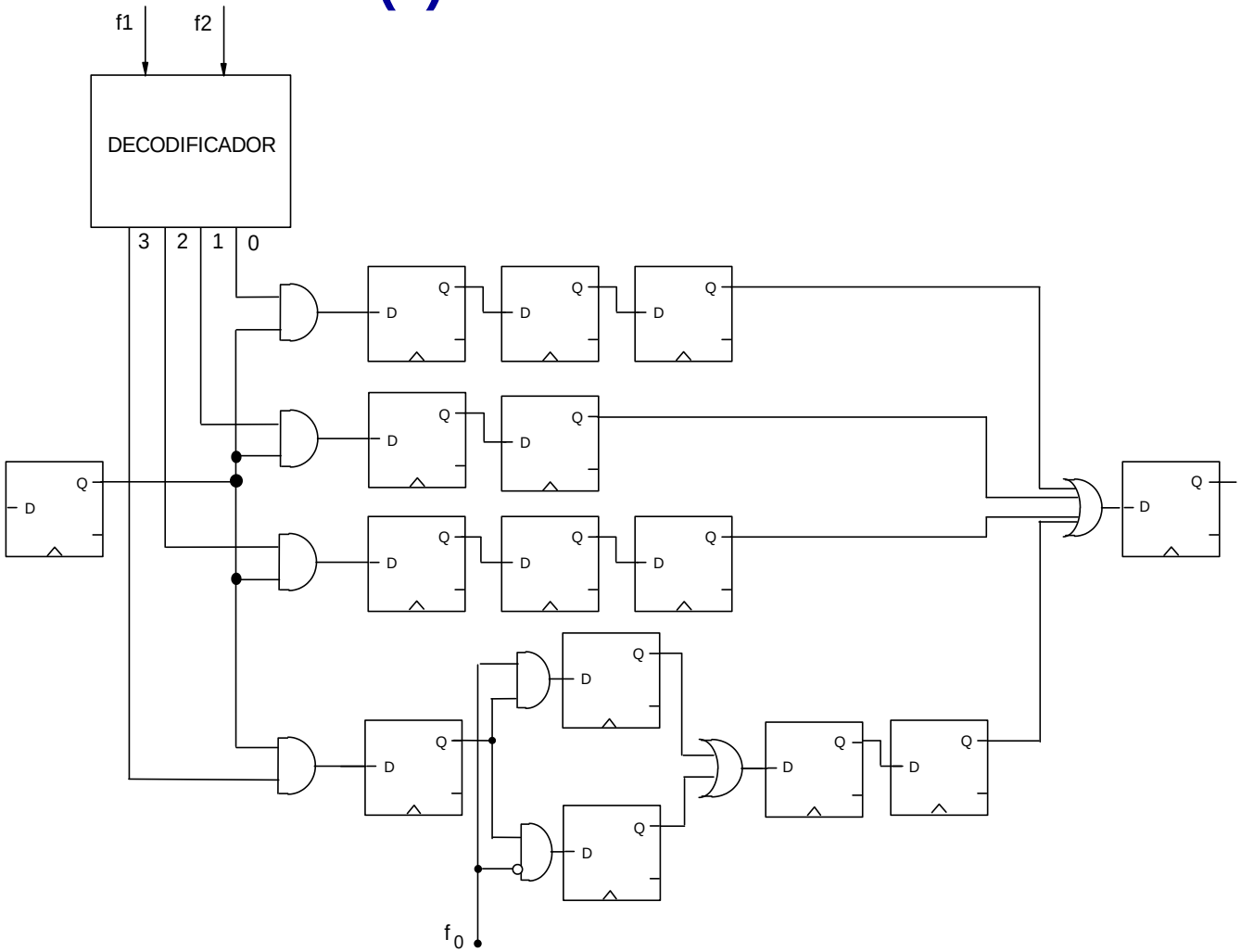
3.1. Control cableado

- **Células de retardo (I)**





- **Células de retardo (II)**



La unidad de control

3. El diseño de la UC

3.2. Control microprogramado

- Consiste en almacenar las distintas "palabras" de control (conjunto de señales de control) correspondientes a cada periodo o fase de ejecución de una instrucción en una memoria

- De esta forma, generar señales de control es leer posiciones de memoria

- Cada "palabra" de control se llama microinstrucción y por eso las UC diseñadas de esta forma se conocen como UNIDADES DE CONTROL MICROPROGRAMADAS

La unidad de control

3. El diseño de la UC

3.2. Control microprogramado

	MICROINSTRUCCIONES							
Periodo / Señal de control	Periodo 1	Periodo 2	Periodo 3	Periodo 4	Periodo 5	Periodo 6	Periodo 7	Periodo 8
MEM	0	1	0	0	0	0	0	0
MR	0	1	1	0	0	0	0	0
MW	0	0	0	0	0	0	0	0
ALE	1	0	0	0	0	0	0	0
MR	0	1	1	0	0	0	0	0
LR	0	0	0	0	0	0	0	1
S0...S3	0	0	0	0	1	0	1	0
X0, X1	0	0	0	0	1	0	1	0
Y0, Y1	0	0	0	0	1	0	1	0
LAC	0	0	0	0	1	0	1	0
DT	0	0	0	0	0	0	0	1
AT	0	0	0	0	0	1	0	0
LPC	0	0	0	0	0	1	0	0
DBT	0	0	0	0	0	0	0	0
PCT	1	0	0	0	0	0	0	0
LI	0	0	1	0	0	0	0	0
IT	0	0	0	0	1	0	0	0
DIR. A	0	0	0	0	0	0	1	1
DIR. B	0	0	0	0	0	0	1	0
LFlags	0	0	0	0	0	0	0	1
RESET	0	0	0	0	0	0	0	1
	fetch			instrucción & CP				

Instrucción

ADD A, B

La anchura de la palabra de control es muy grande ya que ha de gobernar numerosas señales de control

- **Un conjunto de microinstrucciones es un microprograma encargado de ejecutar una determinada instrucción**
- **El conjunto de microprogramas que ejecutan las instrucciones se llama *firmware* o microcódigo**

➔ **En el ejemplo, las cadenas de 1's y 0's correspondientes a:**

➔ periodos 1, 2 y 3 ➔ μprograma de búsqueda (*fetch*)

➔ periodos 5 y 6 ➔ μp de actualización del CP

➔ periodos 4, 7 y 8 ➔ μp de la instrucción ADD

- Las señales de flanco se tratan de la misma forma que las de nivel

- **Una UC microprogramada ha de ser capaz de 3 cosas:**
  1. **Retener en memoria todas las microinstrucciones posibles**
     - ➔ si se suponen instrucciones de código de operación de 7 bits y un contador de periodos de 5 bits sobre 150 señales de control significa un tamaño de memoria de:
     - ➔ 4.096 palabras x 150bits/palabra = 614.400 bits = 600 Kbits
  2. **Hacer corresponder cada instrucción con su microprograma, es decir, hacer que cada código de operación encuentre la dirección donde comienza en memoria el microprograma**
  3. **Seguir la secuencia del microprograma y encadenar con el siguiente código de operación**

- **El primer requerimiento lleva asociado un problema de tamaño de la memoria (coste *hardware*)**

  ➔ **SOLUCIÓN: reducir el formato de la microinstrucción mediante codificación**

- **Los otros dos son problemas de secuenciamiento**

➔ **Las microinstrucciones correspondientes a la ejecución de la instrucción ADD A, B, están formadas básicamente por cadenas de 0's**

➔ Realizan muy pocas operaciones elementales por ciclo (habitualmente 1 única) ya que muchas señales de control realizan tareas incompatibles entre si

- Por ejemplo, accesos simultáneos al mismo bus desde diferentes dispositivos de almacenamiento

- **Esto significa que se está desperdiciando mucha memoria**
  - De todas las posibles combinaciones solamente unas pocas tienen significado real (hacen algo útil)
  - Una palabra que tiene muchos ceros es candidata a la compresión

- **Codificando las microinstrucciones se puede ahorrar memoria de control**

- **Se denomina microprogramación horizontal si no se usa codificación mientras que llamaremos microprogramación vertical al caso de microinstrucciones codificadas**
  - La solución vertical absoluta construye una tabla de palabras de control con sentido y les asigna un código; esta es la solución de formato mínimo
  - La solución real está a medio camino entre la horizontal y la vertical

- **Las microinstrucciones horizontales presentan formatos largos pero son rápidas mientras que las verticales reducen el tamaño de la memoria pero son lentas ya que necesitan decodificación**

La unidad de control

3. El diseño de la UC

3.2. Control microprogramado

- **Las señales de control se pueden agrupar en campos:**
  - Señales que gobiernan el acceso al bus de datos
  - Señales que gobiernan el acceso al bus de direcciones
  - Señales de gobierno de la ALU
  - Señales de gobierno del banco de registros
  - Señales de gobierno de la memoria
  - Señales de gobierno de la unidad de direccionamiento

acceso bus de datos	acceso bus direcciones	gobierno ALU	banco de registros	memoria	unidad de direccionamiento
MR DT IT	AT PCT DBT	S0, ...S3 X0, X1 Y0, Y1 LAC	LR DIR. A DIR. B	MEM MR MW ALE	LPC

- Ya que se puede hacer una clasificación natural de las señales de control, el formato lógico será aquel en el que se respeten estos campos

FORMATO DE MICROINSTRUCCIÓN

bus datos	direcciones	ALU	registros	memoria	U. DIR.

- Una técnica sencilla para ahorrar memoria es codificar los campos; algunos de ellos lo admiten muy fácilmente ya que solamente una señal de control de dicho campo puede estar activa en un periodo: es el caso de los accesos a los buses

- **Señales excluyentes**
  - Cuando un campo contiene señales que no se pueden activar simultáneamente podemos codificar con unos pocos bits los estados posibles

- **Solapamiento de campos**
  - Existen campos que no se pueden activar simultáneamente nunca y, por tanto, pueden solaparse en el formato

- **Mezclar control cableado y control microprogramado**
  - Decodificar directamente los registros involucrados en una operación de proceso desde el registro de instrucción
  - Leer la operación de ALU del registro de instrucción

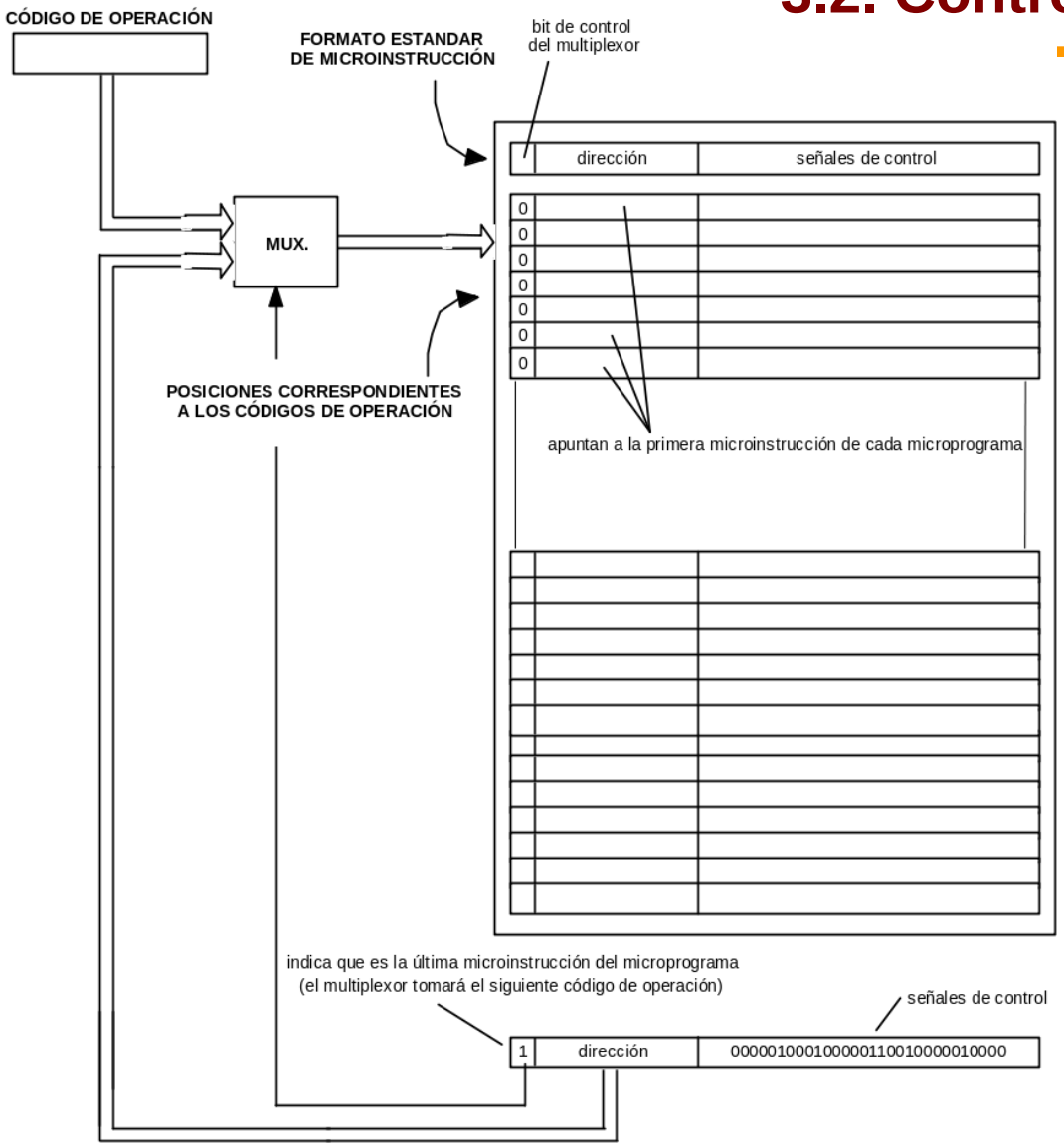
- **Existen dos posibles soluciones:**
  - Secuenciamiento explícito; y
  - Secuenciamiento implícito.

- **Secuenciamiento explícito:**
  - ➔ **Consiste en incluir en cada microinstrucción la dirección de la siguiente**
  - ➔ **Los códigos de operación apuntarían a las primeras posiciones de memoria en las cuales se inicia la secuencia de cada microprograma**
  - ➔ **Cada microinstrucción incluye un bit que indica si es la última microinstrucción o no**

La unidad de control

3. El diseño de la UC

3.2. Control microprogramado



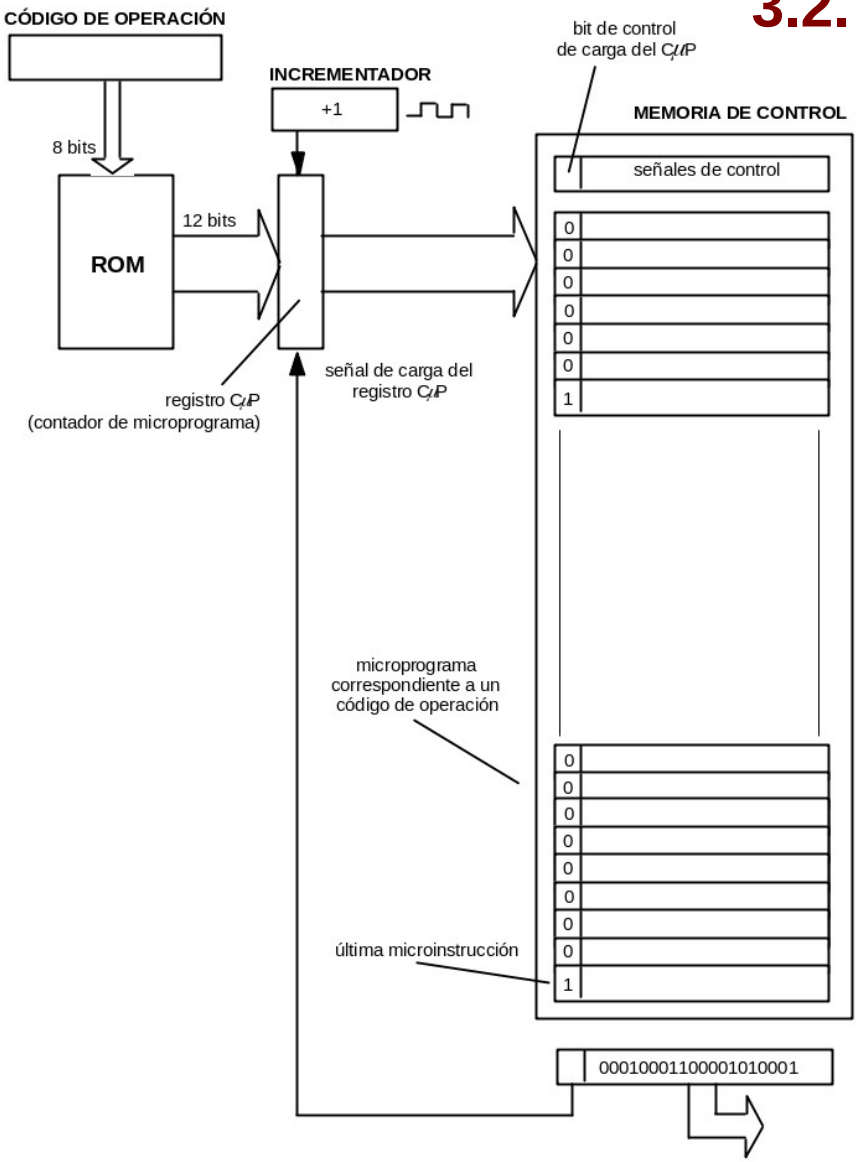
- **Secuenciamiento explícito:**
- **Desventajas:**
  - ➔ **El mayor inconveniente está en la gran cantidad de memoria que se emplea:**
    - ➔ Para 4096 microinstrucciones debo incluir en cada microinstrucción un campo de 12 bits que dé la dirección de la siguiente, es decir, debo emplear 4096 x 12 bits (4Kbit x 12 = 49152bits = 48Kbits) más
- **Ventajas:**
  - ➔ **Permite reutilizar secuencias de microprogramas siempre que sean finales… sin necesidad de escribirlas varias veces (ahorro de memoria)**

- **Secuenciamiento implícito:**
  - ➔ **Las microinstrucciones correspondientes a un microprograma están ordenadas secuencialmente**
  - ➔ **Un contador de microprograma (CP) va apuntando a las sucesivas microinstrucciones**
  - ➔ **Para encontrar la primera microinstrucción de cada microprograma se usa una tabla**
    - ➔ La tabla es una ROM indexada por los códigos de operación de las distintas instrucciones y cuyo contenido es la posición de la primera microinstrucción de cada microprograma

La unidad de control

3. El diseño de la UC

3.2. Control microprogramado



- **Microbifurcaciones condicionales**
  - ➔ **Las instrucciones de salto condicional tienen dos cronogramas posibles, es decir, poseen dos microprogramas diferentes que se ejecutan dependiendo de la condición**
  - ➔ **Necesito un mecanismo de microsalto o microbifurcación que seleccione la ejecución de un microprograma u otro: una determinada microinstrucción debe poder elegir entre dos direcciones o caminos alternativos**

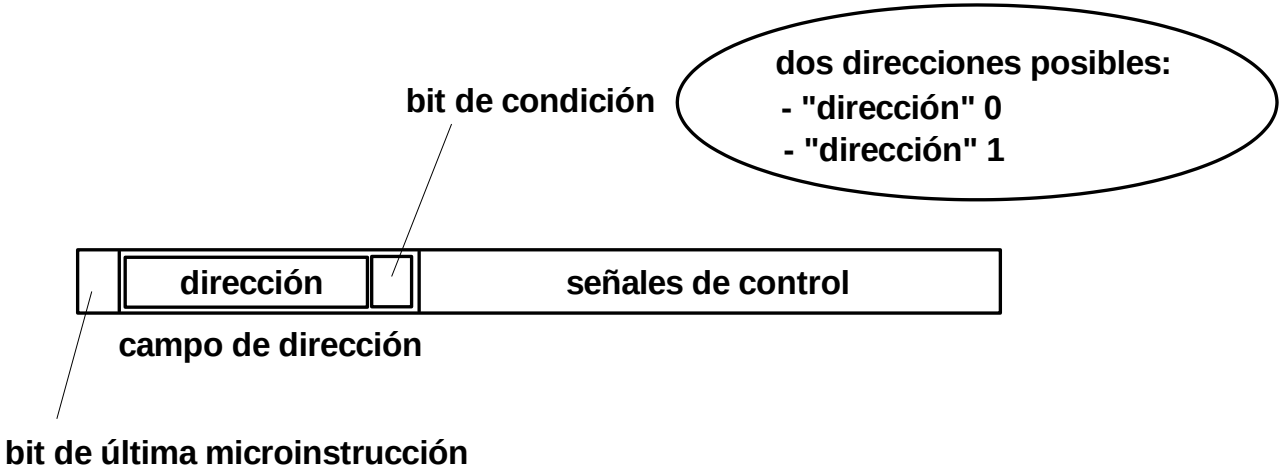
- **El mecanismo dependerá del tipo de secuenciamiento**

- **En el secuenciamiento explícito:**
  - Cada microinstrucción lleva la dirección de la siguiente
  - Incluir dos direcciones significa agrandar demasiado el formato por lo que hay que intentar dar una sola... o dos que difieran en un solo bit, por ejemplo, dos direcciones consecutivas
  - El bit en que difieren será el resultado de la comparación con la condición: 1 si se cumple y 0 si no se cumple
  - De esta forma se accede a dos microinstrucciones distintas que llevan por caminos alternativos

La unidad de control

3. El diseño de la UC

3.2. Control microprogramado

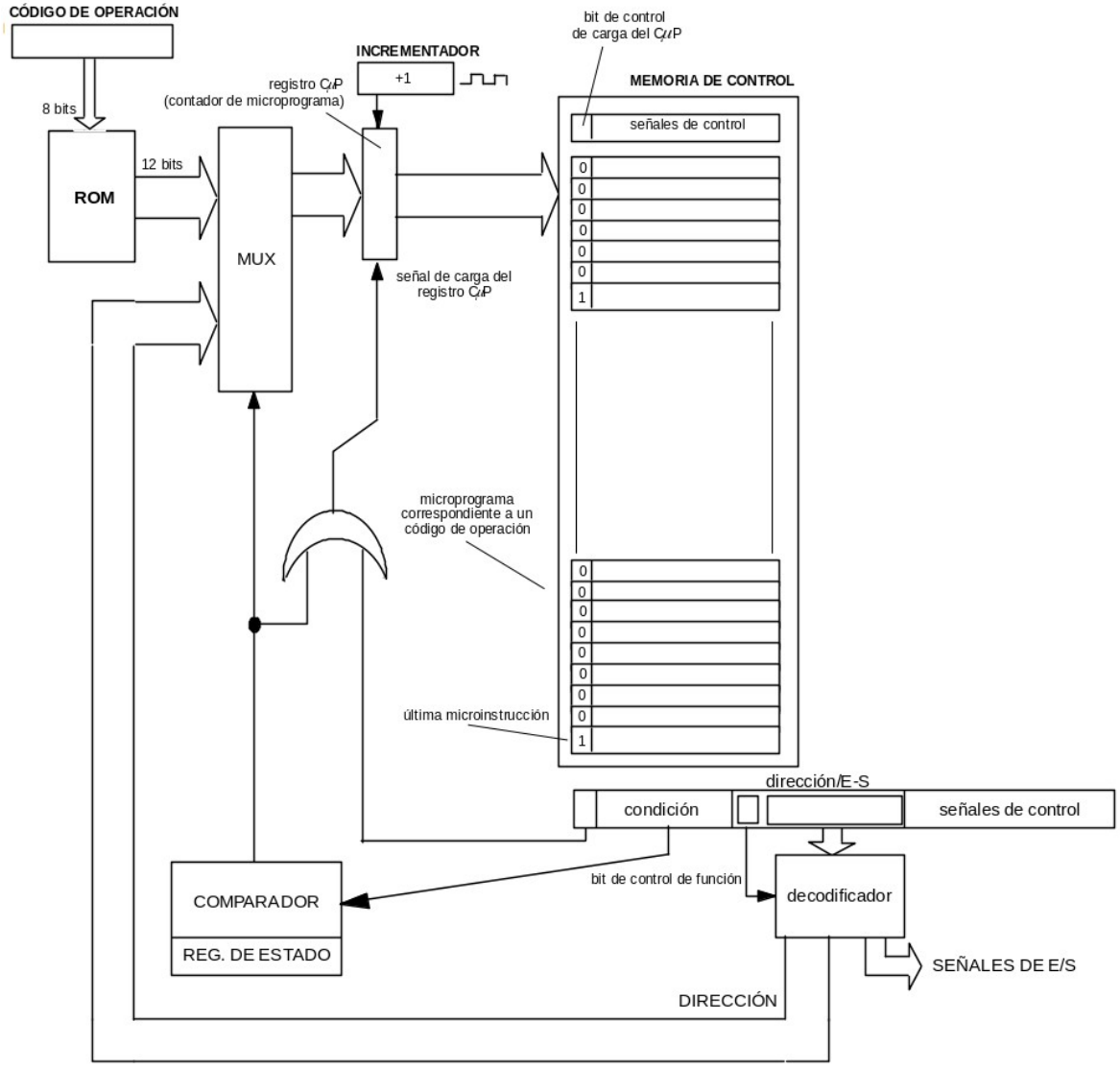


- **En el secuenciamiento implícito:**
  - No se puede incluir un campo para esta dirección ya que estaríamos en el secuenciamiento explícito...
  - La solución consiste en **solapar el campo** de esta dirección con un campo de función excluyente, es decir, un campo que tenga una función que no se dé nunca a la vez con la especificación de una dirección en las microinstrucciones del microprograma de salto. Este es el caso,por ejemplo, de las E/S: los saltos nunca realizan a la vez operaciones de E/S

La unidad de control

3. El diseño de la UC

3.2. Control microprogramado



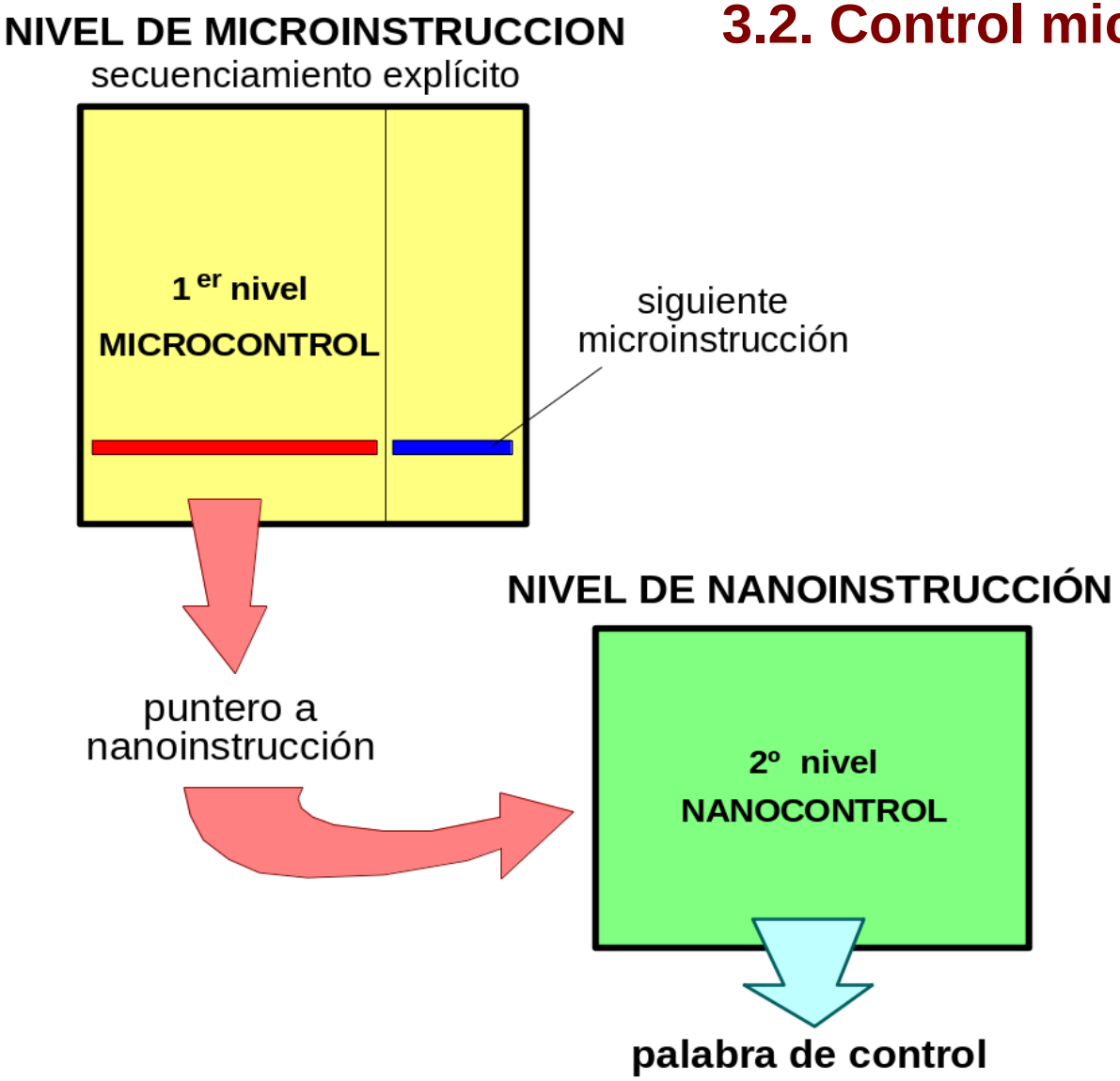
- **El MC68000 cuenta con una unidad de control microprogramada en dos niveles**
  - Reduce el coste hardware

- **En una estructura de dos niveles**
  - El primer nivel (microcontrol) contiene secuencias de palabras de control que son punteros (nanodirecciones) al segundo nivel
  - El segundo nivel contiene una ordenación arbitraria de palabras de control NO DUPLICADAS

La unidad de control

3. El diseño de la UC

3.2. Control microprogramado



➔ **La eficiencia de esta estructura descansa en dos hechos:**

1. El número de palabras de control implementadas debe ser una fracción pequeña del número total de las posibles
   - El MC68000 tiene 70 señales ➔ 2^{70} combinaciones
   - El conjunto completo de instrucciones de ensamblador de este procesador no necesita más de 200 ó 300 palabras
   - Este conjunto puede ser direccionado con 9 bits

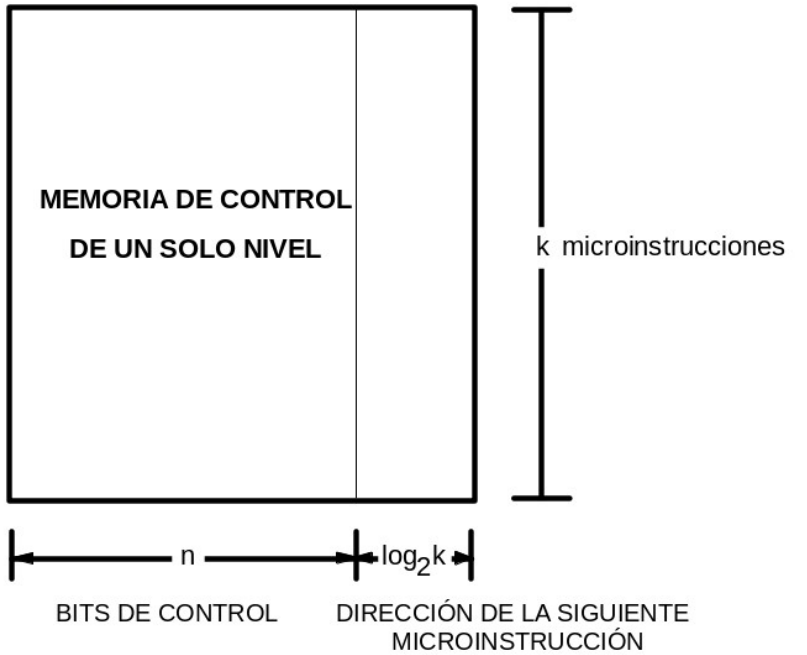
Solamente unas pocas palabras de control tienen significado

2. Debe de darse una cierta redundancia en el uso de las nanoinstrucciones
   - Si la correspondencia fuera 1 a 1 la dirección del primer nivel sería sustituida por la palabra del control de la nanoinstrucción (eliminando el 2º nivel)
   - Los punteros se repiten 2 ó 3 veces de forma que lo repetido es de menor tamaño (9 bits frente a 70)

Hay una cierta reutilización de palabras de control en diferentes instrucciones

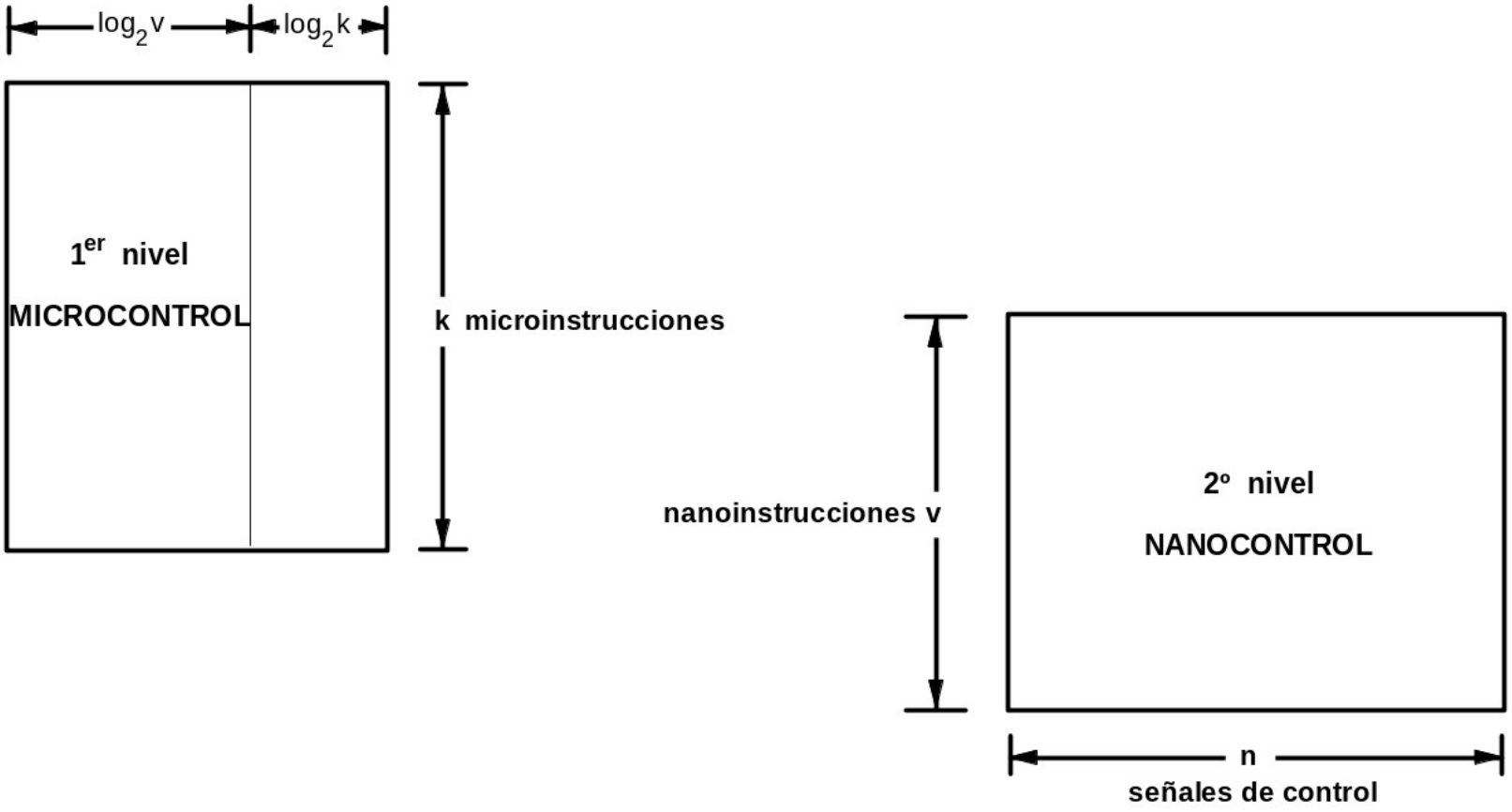
- **Comparativa de tamaños entre microprogramación y nanoprogramación**
  - Sean las siguientes variables:
  - *n*: número de señales de control diferentes en la arquitectura de un procesador
  - *k*: número total de microinstrucciones (palabras de control) necesarias para implementar todas las instrucciones de ensamblador
  - *p*: relación entre el número de microinstrucciones diferentes y *k*

- **Tamaño de una unidad de control microprogramada**
  - Secuenciamiento explícito
  - Si *k* es el número de microinstrucciones y cada una de ellas contiene una palabra de control de *n* bits y una dirección de la siguiente microinstrucción, tendremos el siguiente tamaño:



$$T_1 = k \cdot (n + \log_2 k)$$

➔ **Tamaño de una unidad de control microprogramada en dos niveles**



- En el 1er nivel se almacena una dirección que apunta al 2º nivel en el que se han escrito las palabras de control no repetidas
- El ancho de la nanodirección vendrá dado por el número de palabras de control diferentes:

  (1)
  $$v = \rho \cdot k$$

- El tamaño del primer nivel será el ancho de la nanodirección más el de la microdirección:

  (2)
  $$k \cdot \left(\log_2 v + \log_2 k \right)$$

- El tamaño del segundo nivel vendrá dado por el número de palabras de control no repetidas (nanoinstrucciones) y el número de señales de control que contienen:

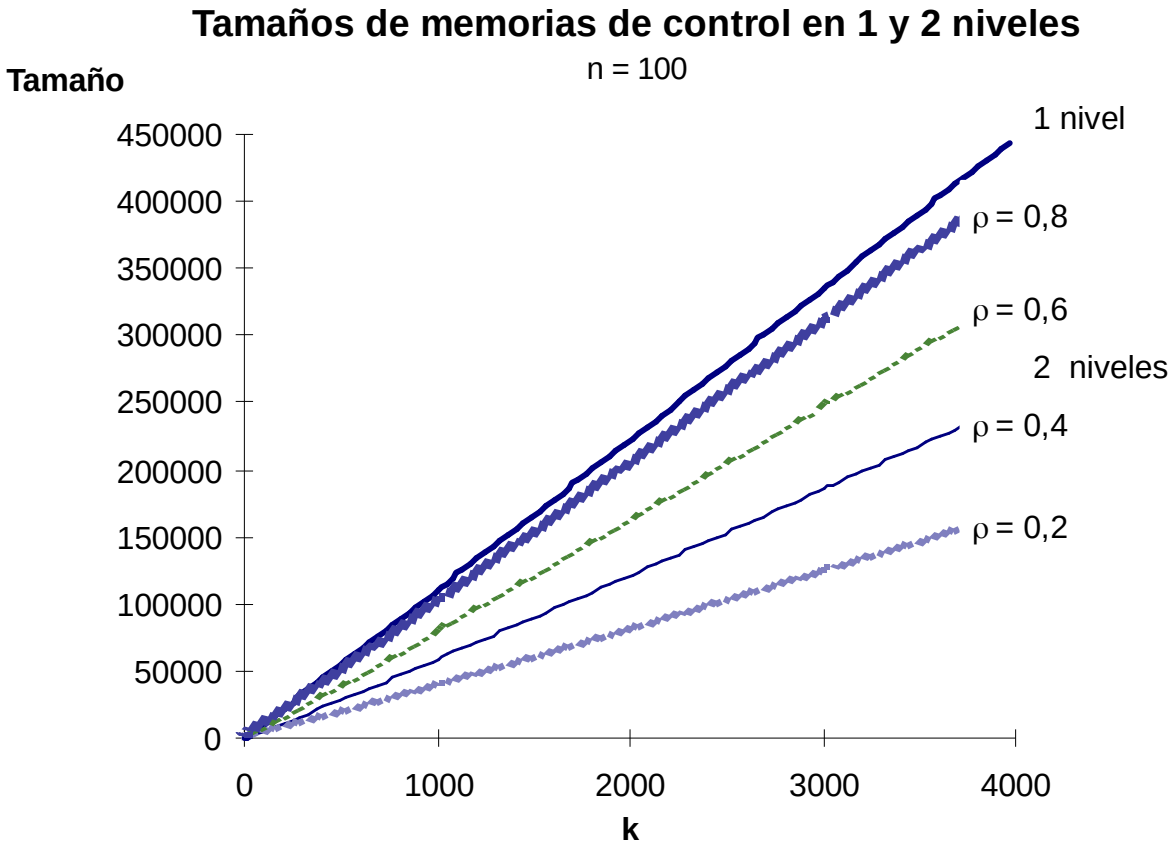
  (3)
  $$n \cdot v$$

- El tamaño total será la suma de las ecuaciones (2) y (3):

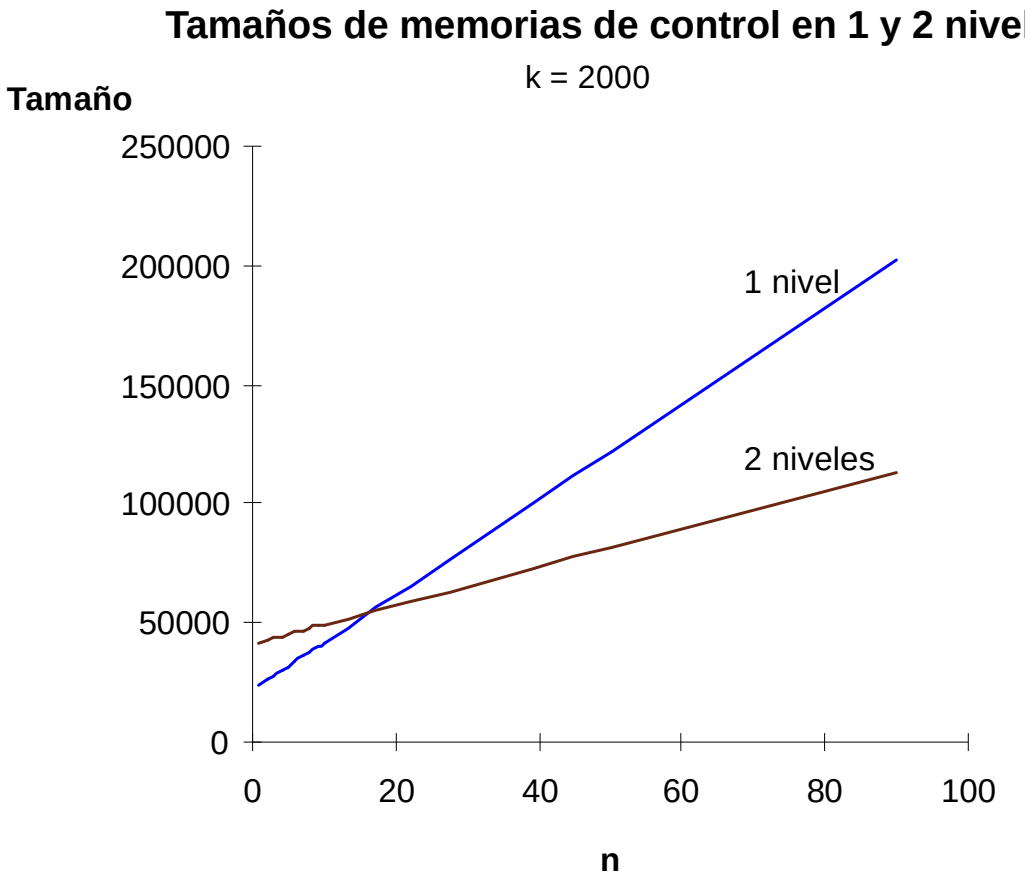
  (4)
  $$T_2 = k \cdot \left(\log_2 v + \log_2 k \right) + n \cdot v$$

- **Comparación de tamaños**
  - El objetivo es que T2 sea menor que T1. ¿Para que valores de *n*, *k* y es cierto esto? En primer lugar debemos tener una idea de los rangos en los que se mueven estos parámetros para procesadores reales
  - Tenemos que:
    - *n* (20, 750)
    - *k* (50, 8500)

➔ **Comparativa de tamaños de UC microprogramada en un nivel y en dos niveles para diferentes valores de *ρ* en función de *k* para *n* fijo**



➔ **Comparativa de tamaños de UC microprogramada en un nivel y en dos niveles en función de *n* para *k* fijo**



NANOPROGRAMACIÓN EN EL MC68000

- **Para el procesador MC68000 tenemos que:**
  - $n = 70$
  - $k = 650$
  - $\rho = 0{,}4$
  - $v = 260$
  - de donde:
  - $T1 = 650 \cdot (70 + \log_2 650) = 52.400$ bits
  - $T2 = 650 \cdot (\log_2 260 + \log_2 650) + 70 \cdot 260 = 30.550$ bits
  - $T2 / T1 = 0{,}58$
  - $T = T1 - T2 = 21.850$ bits