Tema 3: Procesador

- Índice
  - Introducción
  - Operaciones elementales
  - Estructura de un computador elemental y sus señales de control
  - Cronogramas
  - Diseño de la UC
  - UC microprogramada
  - Control de excepciones

Objetivos

- Visión dinámica del computador:
  - mostrar cómo se ejecutan las instrucciones
  - describir el órgano encargado de que esto se lleve a cabo (la unidad de control)
  - entender esta ejecución de instrucciones mediante su representación en el tiempo por cronogramas
- Comprender el diseño de la unidad de control
  - Ser capaz de desglosar las instrucciones en operaciones básicas según la estructura del computador
  - Diseño de la unidad de control microprogramada
- Comprender y analizar aspectos de diseño que pueden mejorar el rendimiento a través de mejoras en la estructura del computador o de la UC

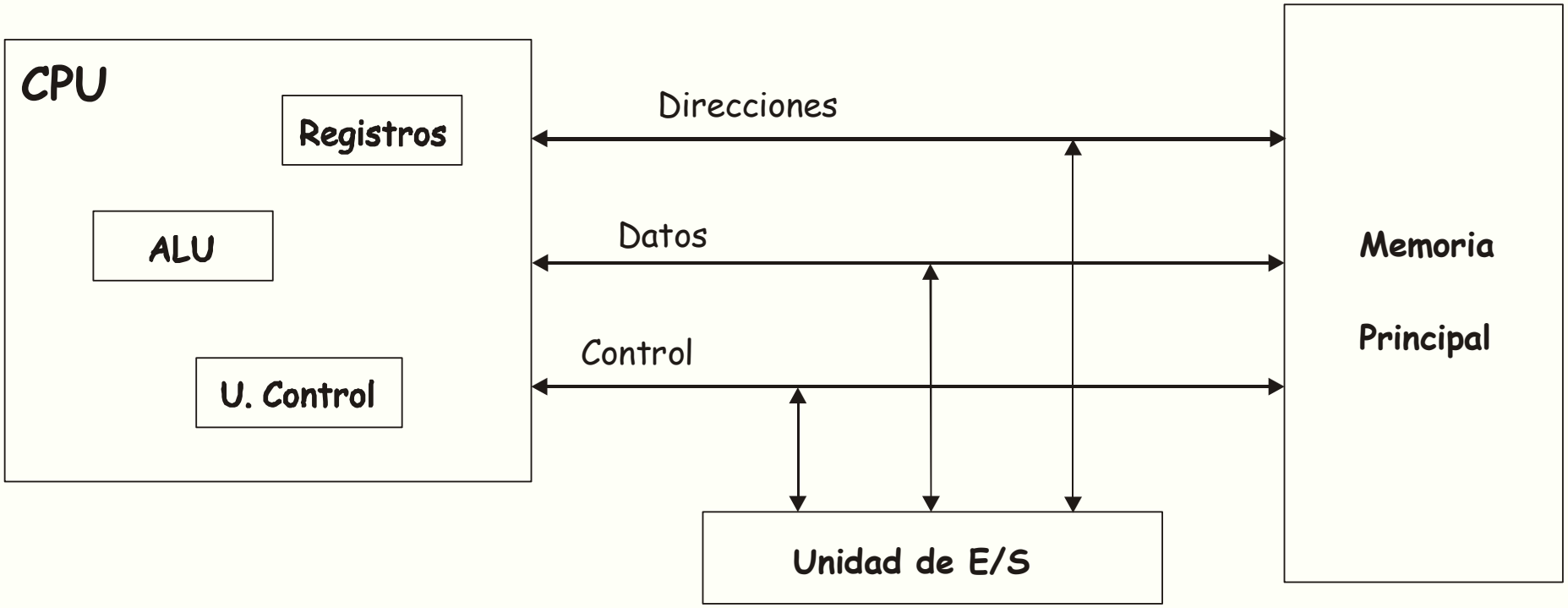
Bibliografía

- de Miguel, P. *"Fundamentos de los computadores",* Paraninfo, 2004. 9ª edición
- Patterson, D. A.; Hennessy, J. L. *Computer Organization and Design.* Morgan-Kaufmann. 2011. 4ª edición
- Stallings, W. *"Organización y arquitectura de computadores",* Prentice Hall, 2006, 7ª Edición

Introducción

- Estudiaremos:
  - **Unidad de control**:
    - Encargada de interpretar las instrucciones del programa y gobernar la ejecución de las mismas
  - **Camino de datos**:
    - Conexiones de la CPU por las que se transfieren los datos procedentes de la memoria o registros internos, para obtener los resultados
    - Debe permitir el conjunto de operaciones básicas que precisa el repertorio de instrucciones
- Organización de procesadores: ha evolucionado
  - desarrollo tecnológico
  - la necesidad de obtener altas prestaciones

Esquema básico del computador Von Neumann. Componentes



Introducción

- Funciones de la CPU:
  1. Ejecuta instrucciones (función básica)
     - Lectura, decodificación, e interpretación de las instrucciones
     - Generación de órdenes para la ejecución
     - Secuenciamiento de las instrucciones
       (decidir cuál es la siguiente a ejecutar)
  2. Resuelve situaciones anómalas
     (desbordamiento, operación no válida, error de paridad, etc)
  3. Controla la comunicación con periféricos

Introducción

- Entradas y salidas de la UC:
  - Entradas:
    - Registro de Instrucción: CO, MDs
    - Reloj: registro contador de fases
    - Registro de estado
    - Señales de control externas (E/S, Mem,)
  - Salidas
    - Todas las señales de control que permiten realizar cada una de las instrucciones máquina: internas y externas (E/S, Mem)

Introducción

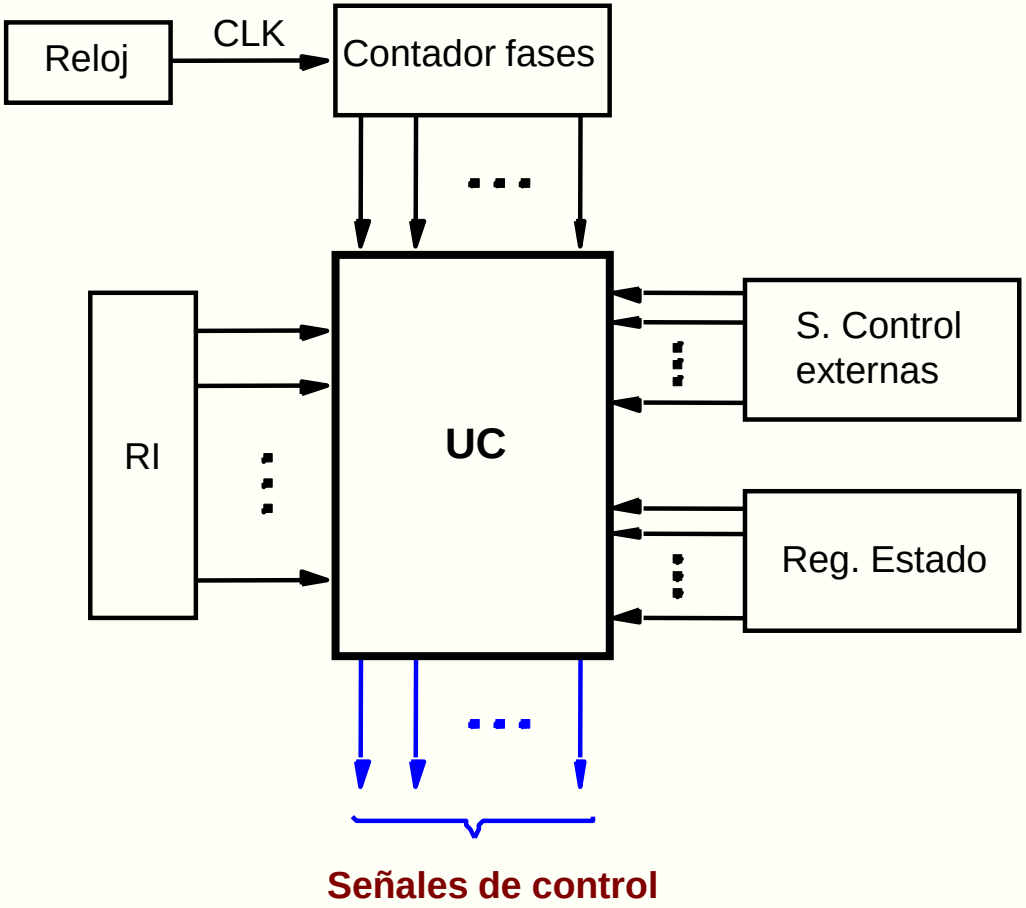
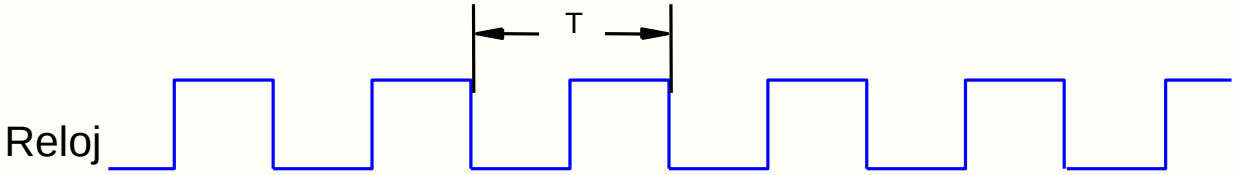


Figura Entradas y salidas de la UC.

Introducción

- **Reloj**: tren de pulsos caracterizado por su periodo



  - ***señales de control***: siempre sincronizadas con el reloj
  - Define el tiempo de cada operación.
    - Memoria: tiempo de lectura y escritura
    - ALU: tiempo de operación
  - **CAMINO CRÍTICO**: camino de máximo retardo entre un origen y un destino. Depende de los dispositivos que tengan que atravesar las señales.

Ej: si f=100 MHz

T= 10 ns y conviene que una operación aritmética se realice en un tiempo algo menor que 10 ns

¿Y Mem?

Introducción

Ciclo de lectura

Se supone AR siempre disponible en el bus de direcciones de memoria

1. dirección → AR
2. M(AR) → Reg; lectura

lectura: MEMRQ y RD
Reg o (DR)

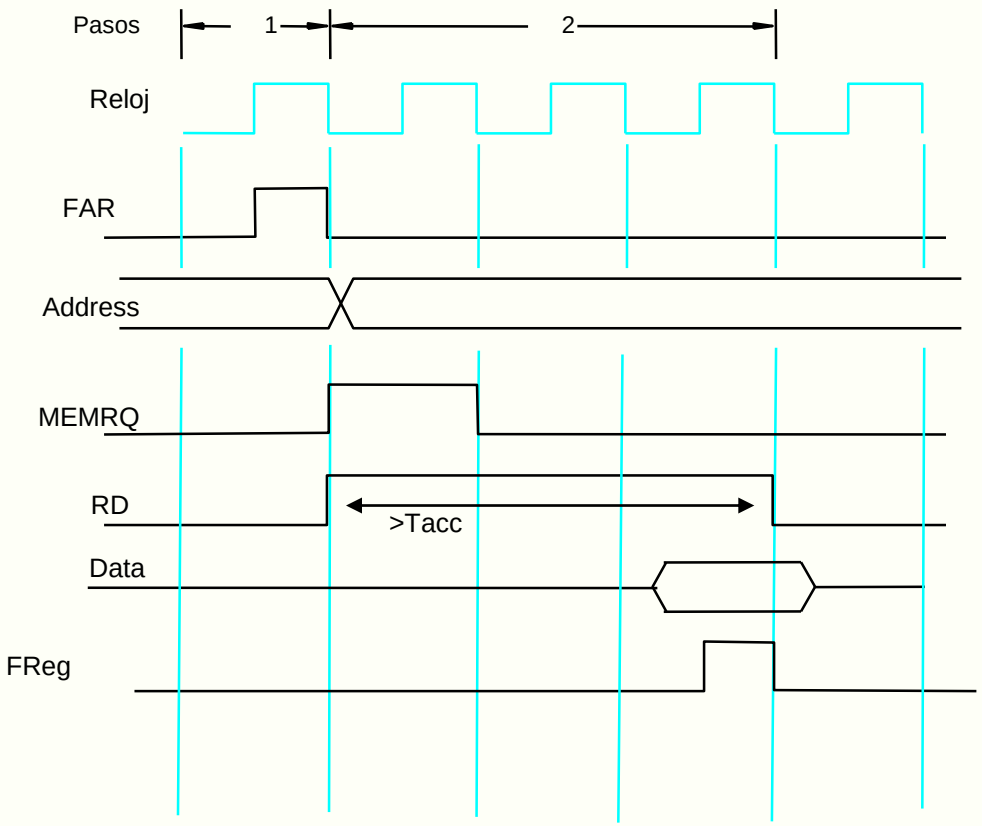


Figura . Temporización de la lectura en memoria

Introducción

Ciclo de escritura

AR y DR disponibles
en buses de dir y dat

1. dirección → AR
2. dato → DR
3. DR → M(AR) ; escritura

escritura: MEMRQ y WR

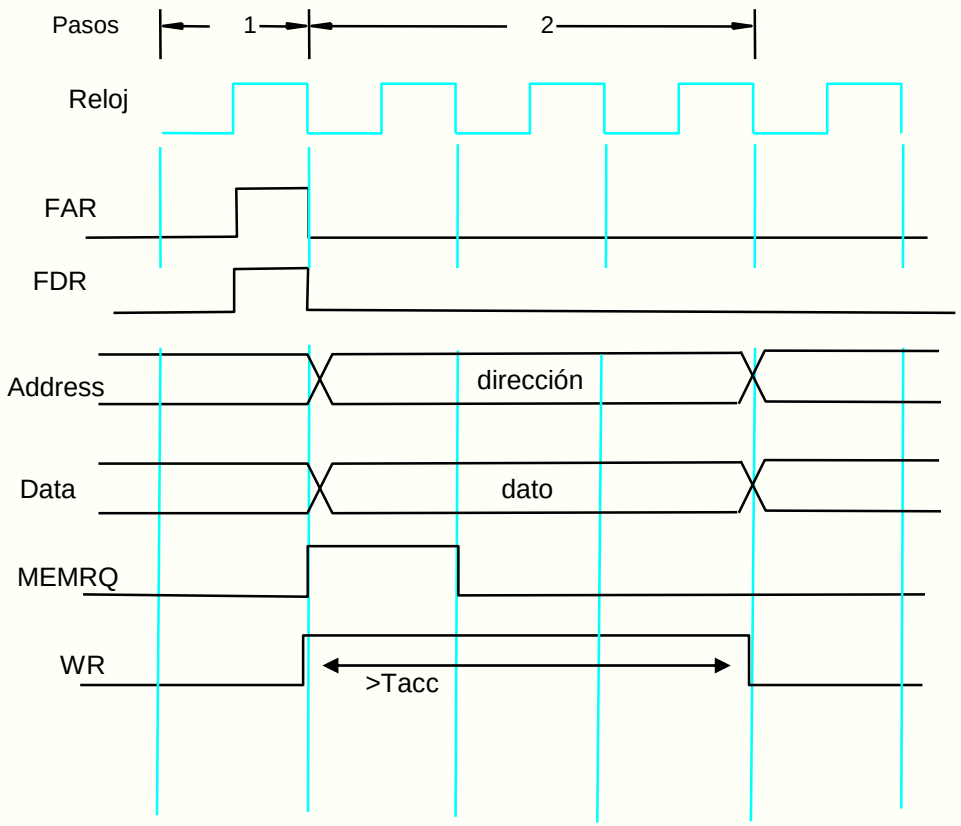


Figura . Temporización de la escritura en memoria

Introducción

Ciclos de lectura y escritura (en Mem): **ciclo de BUS**

- implica un acceso al exterior a la CPU
  - durante ese tiempo sólo la CPU puede acceder a los buses
  - (ningún periférico)
- suele durar más de 1 ciclo de reloj

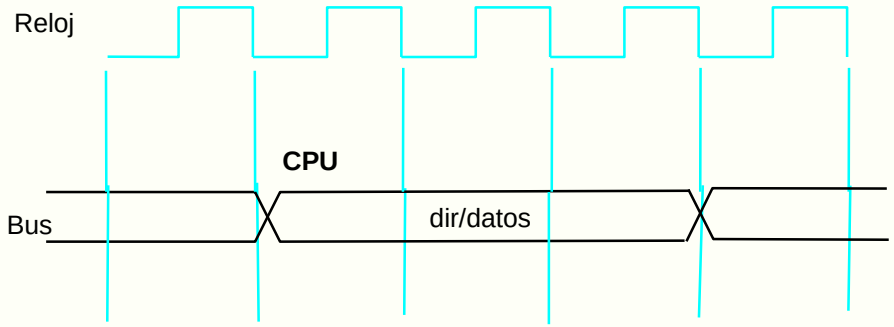


Figura . Ciclo de bus

Introducción

Evaluación del rendimiento

- Supóngase los tiempos de ejecución:

  Acceso a memoria: 8 ns

  ALU y sumadores: 2 ns

  Acceso a registros: 1 ns

- ¿Cúal de las siguientes realizaciones será más rápida?
  - Una 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)
- ¿De qué dependería la duración del periodo de reloj?

Unidad de Control

- Índice
  - Introducción
  - Operaciones elementales
  - Estructura de un computador elemental y sus señales de control
  - Cronogramas
  - Diseño de la UC
  - UC microprogramada
  - Control de excepciones

Operaciones Elementales

- Las **fases de ejecución** ayudan a simplificar el diseño de la UC.

  Ej: fetch común a todas las instrucciones

- Funcionamiento del computador durante la ejecución de un programa consiste en una **sucesión de ciclos de instrucción**

Operaciones Elementales

- **OPERACIONES ELEMENTALES (microoperaciones)**: operaciones (realizables directamente por el hardware) en que la UC divide cada una de las fases de ejecución de una instrucción
- Las operaciones elementales se realizan por la UC mediante la activación de señales de control
- Cada microoperación dura un ciclo de reloj (excepto la Mem)
- Se pueden simultanear en el tiempo, cuidando:
  - Orden prestablecido
  - Conflictos en los elementos Hw

Operaciones Elementales

- Tipos:
  - **De transferencia**

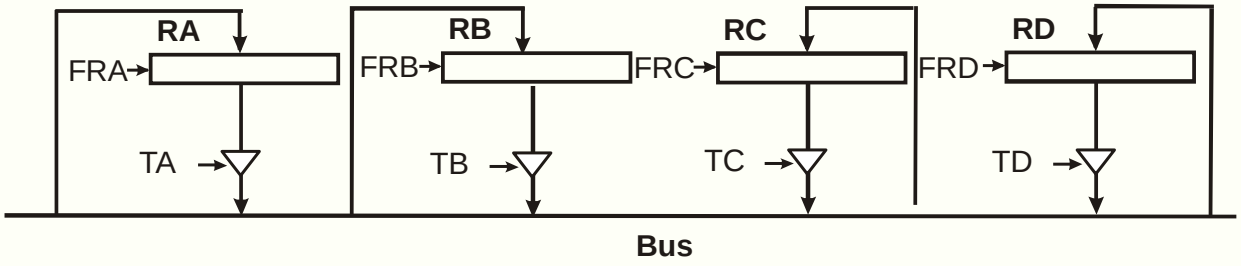
    Llevan información de ORIGEN a DESTINO
  - **De proceso**

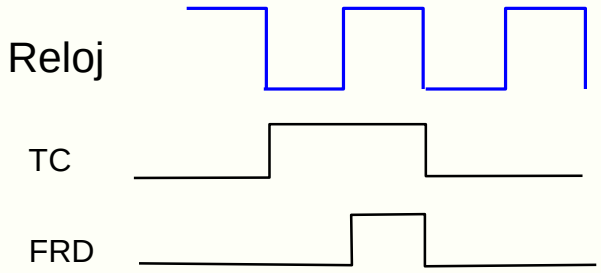
    Llevan información de ORIGEN a DESTINO, pero ésta pasa por operador

Operaciones Elementales

- **OE de transferencia**
  - Llevan información de un ORIGEN a un DESTINO
    1. Establecer camino físico entre salida de origen y entrada de destino
    2. Enviar señal al destino para que tome la información

    Ej: Transferencia a través de un bus de dos registros **C → D**



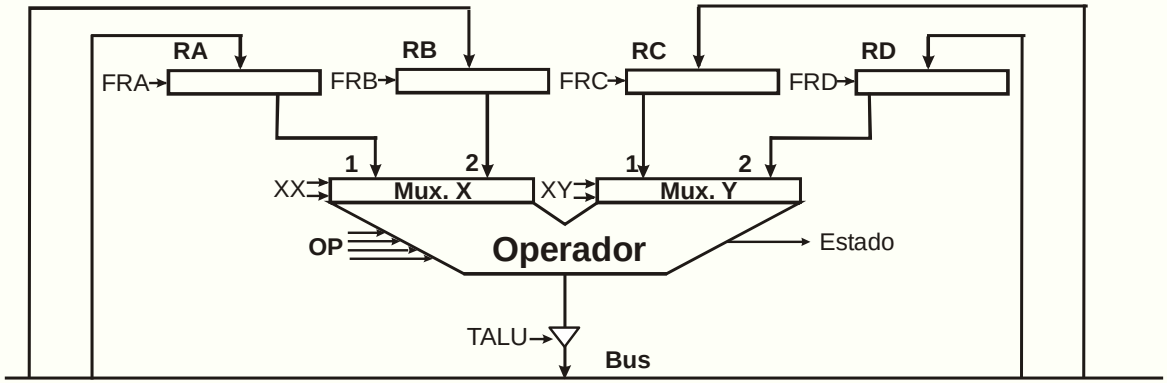


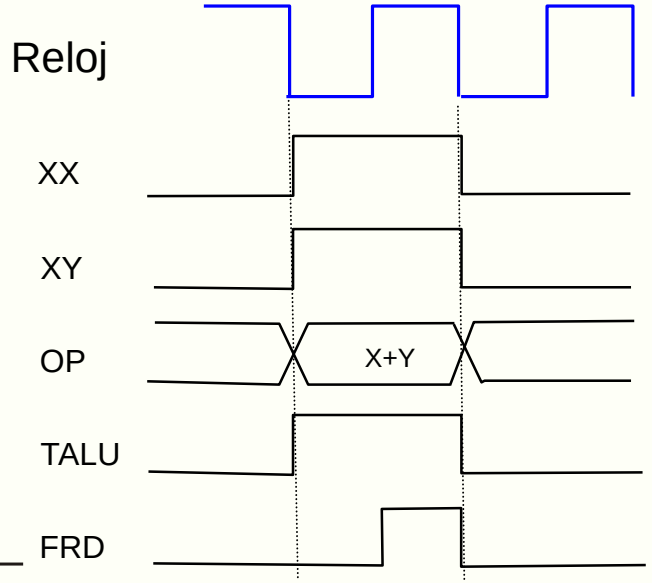
Operaciones Elementales

- **OE de proceso**
  - Funcionamiento similar a la transferencia, pero la información de ORIGEN se pasa por un operador que la procesa en su camino al DESTINO

    Ej: Operación en la ALU con registros

    A+C → D





Operaciones Elementales

- **Reglas de agrupación de μoperaciones**
  - Respetar orden prestablecido de algunas acciones

    Ej: antes de leer, guardar dirección en AR
  - Evitar conflictos en el mismo recurso físico

    un dispositivo no puede estar en dos estados diferentes al mismo tiempo

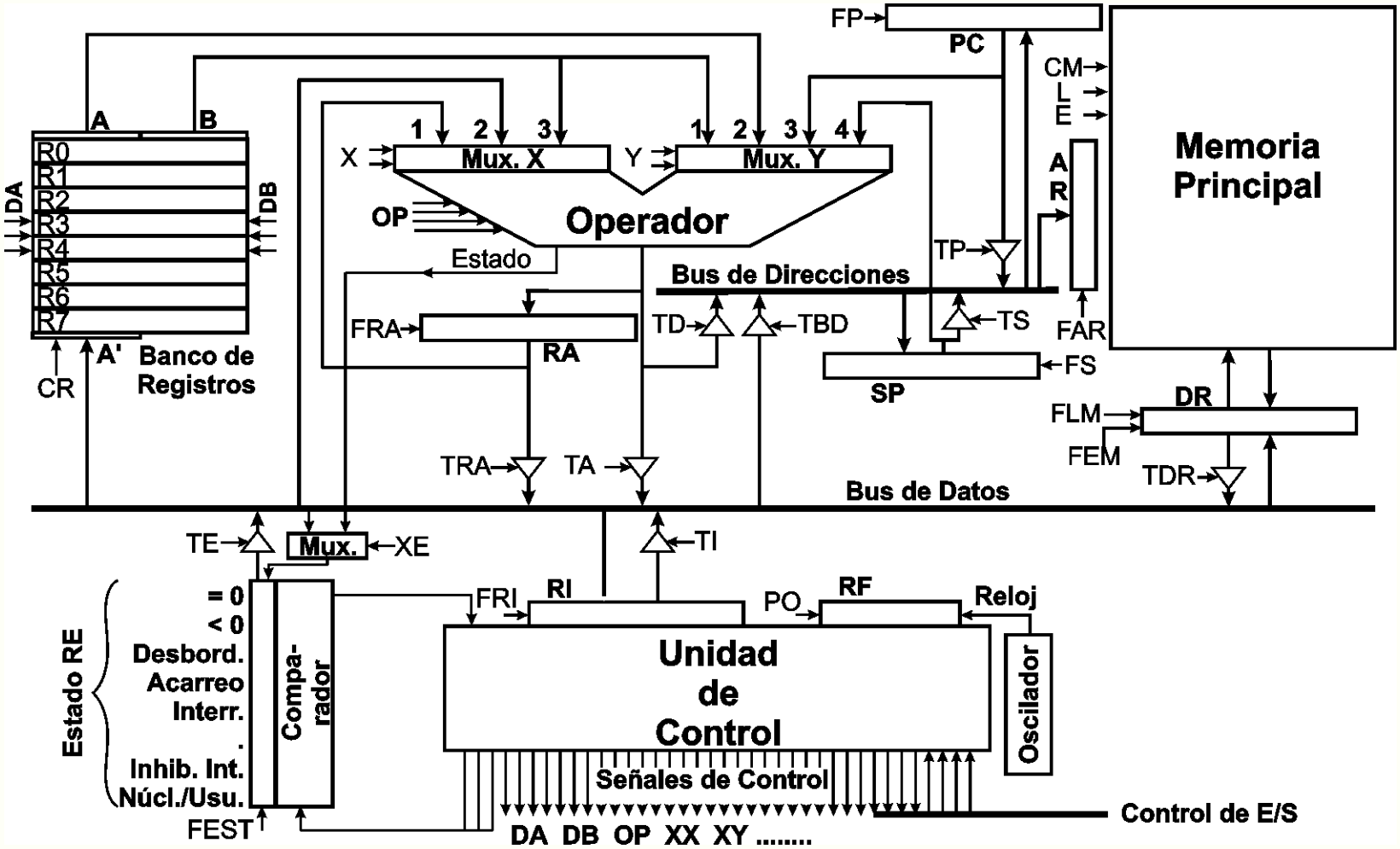
    Ej: conflicto en bus (dos op. elem. intentan usar el mismo bus);

    memoria no puede leer y escribir al mismo tiempo
  - La información debe guardarse en algún sitio: ORIGEN y DESTINO deben poder almacenar información (reg o mem)

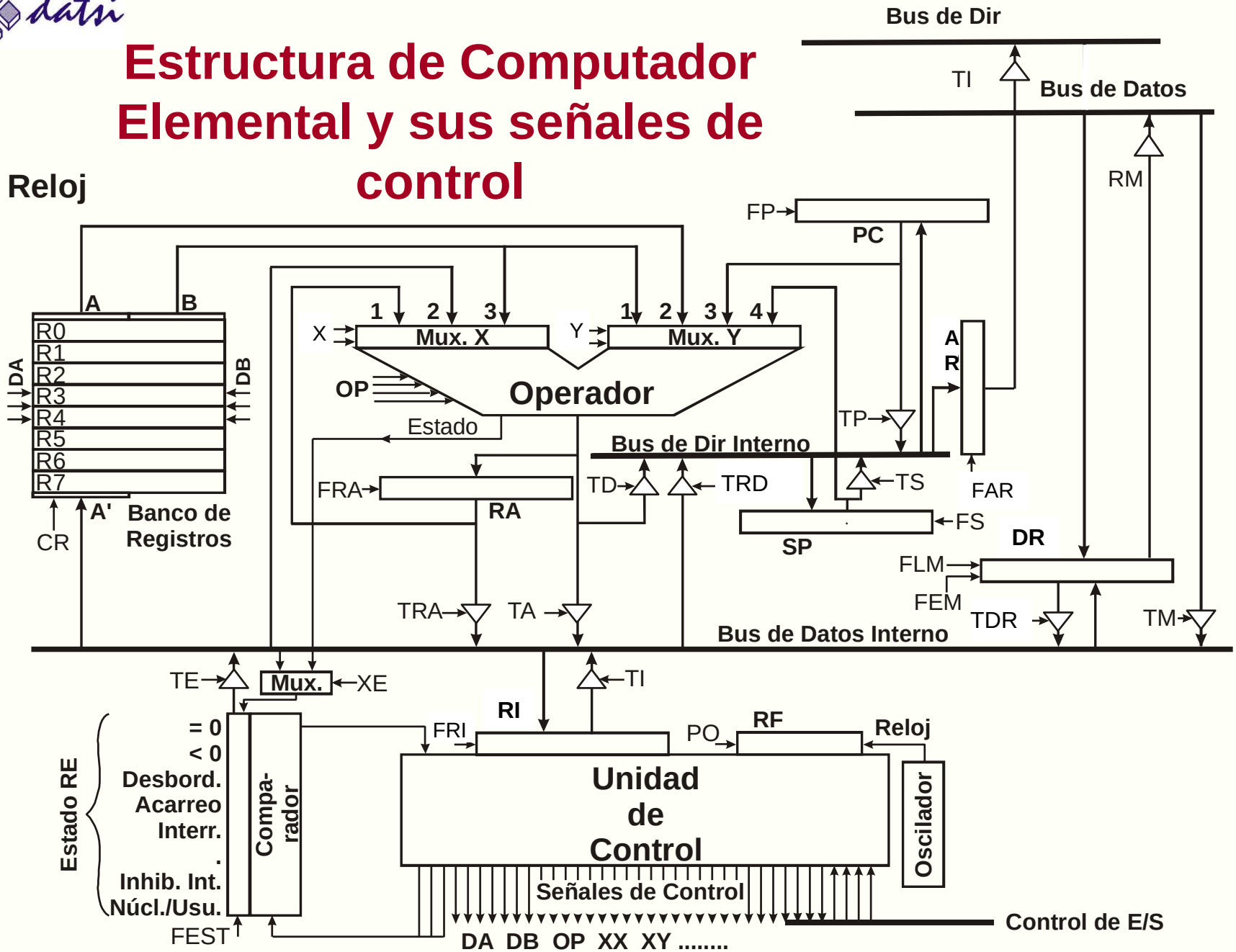
Unidad de Control

- Índice
  - Introducción
  - Operaciones elementales
  - ► Estructura de un computador elemental y sus señales de control
  - Cronogramas
  - Diseño de la UC
  - UC microprogramada
  - Control de excepciones

Estructura de Computador Elemental y sus señales de control



Estructura de Computador Elemental y sus señales de control

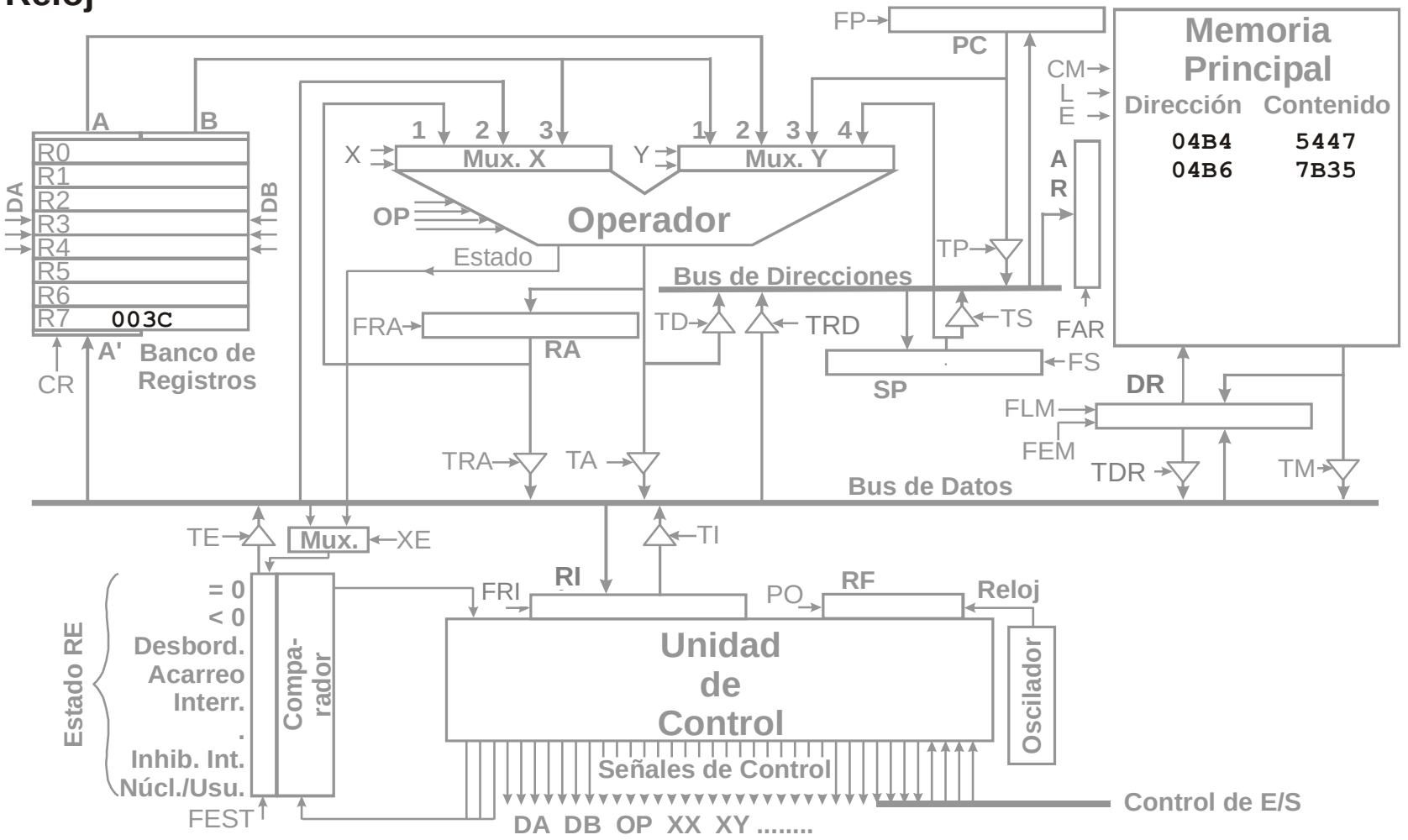


Unidad de Control

- Indice
  - Introducción
  - Operaciones elementales
  - Estructura de un computador elemental y sus señales de control
  - ► Cronogramas
  - Diseño de la UC
  - UC microprogramada
  - Control de excepciones

ADD .R4,.R7

Reloj



Esta animación presenta la forma en que un computador sencillo ejecuta
la instrucción de suma de dos registros.

Se supondrá que la arquitectura es de 16 bits y que la memoria se direcciona a nivel de byte. Además, el acceso a la memoria principal requiere dos ciclos.

La instrucción se encuentra almacenada en la posición de memoria H’4B4, ocupa 16 bits y se compone del código de operación (H’54) seguido de los descriptores de los registros, (4 y 7).

El resultado se dejará en el registro 4.

Los contenidos de los registros son H’4F2 y H’3C, por lo que el resultado es H’52E. Además, el único bit de estado que se activa es de NZ, puesto que el resultado no es cero.

La animación contempla dos imágenes por ciclo de reloj. La primera corresponde a las señales de control de tipo nivel, mientras que la segunda refleja las señales de carga.

Para ver el detalle de un ciclo se recomienda la animación “element”, que muestra el desarrollo de una operación elemental.

Solamente se han indicado los valores de los registros afectados por la instrucción.

Se ha considerado que la lectura de la instrucción empieza en el ciclo 3 de la instrucción anterior y que el contador de ciclos se pone a “0” en el momento de la decodificación de la instrucción leída.

Las operaciones elementales necesarias son las siguientes:

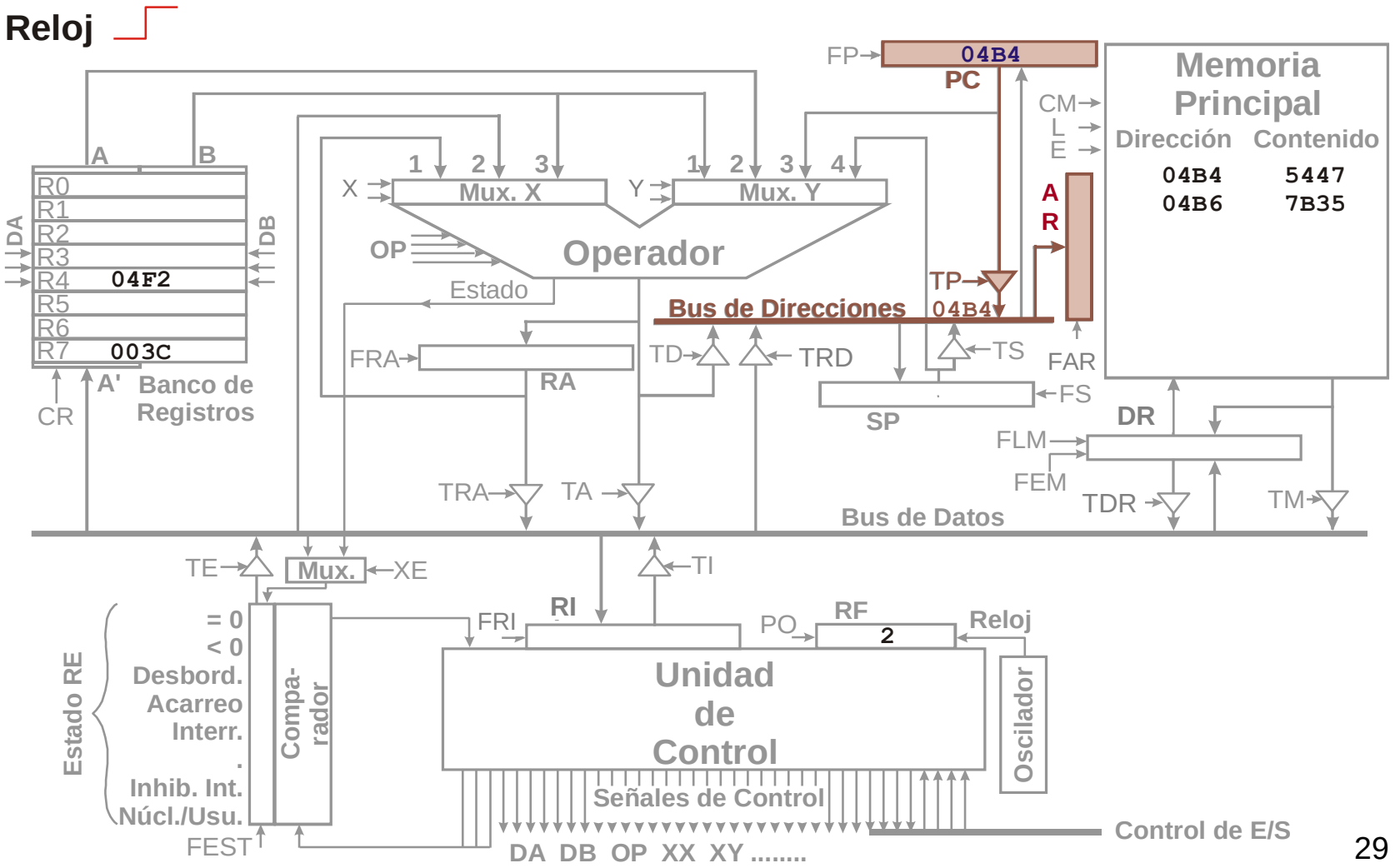
D ← PC
I ← M(D); esta lectura tarda dos ciclos
Se decodifica la instrucción
R4 ← R4 + R7

Además de estas operaciones, se incrementa el contador de programa, para ir preparando la lectura de la siguiente instrucción.

PC ← PC + 2

Prepara la lectura instrucción

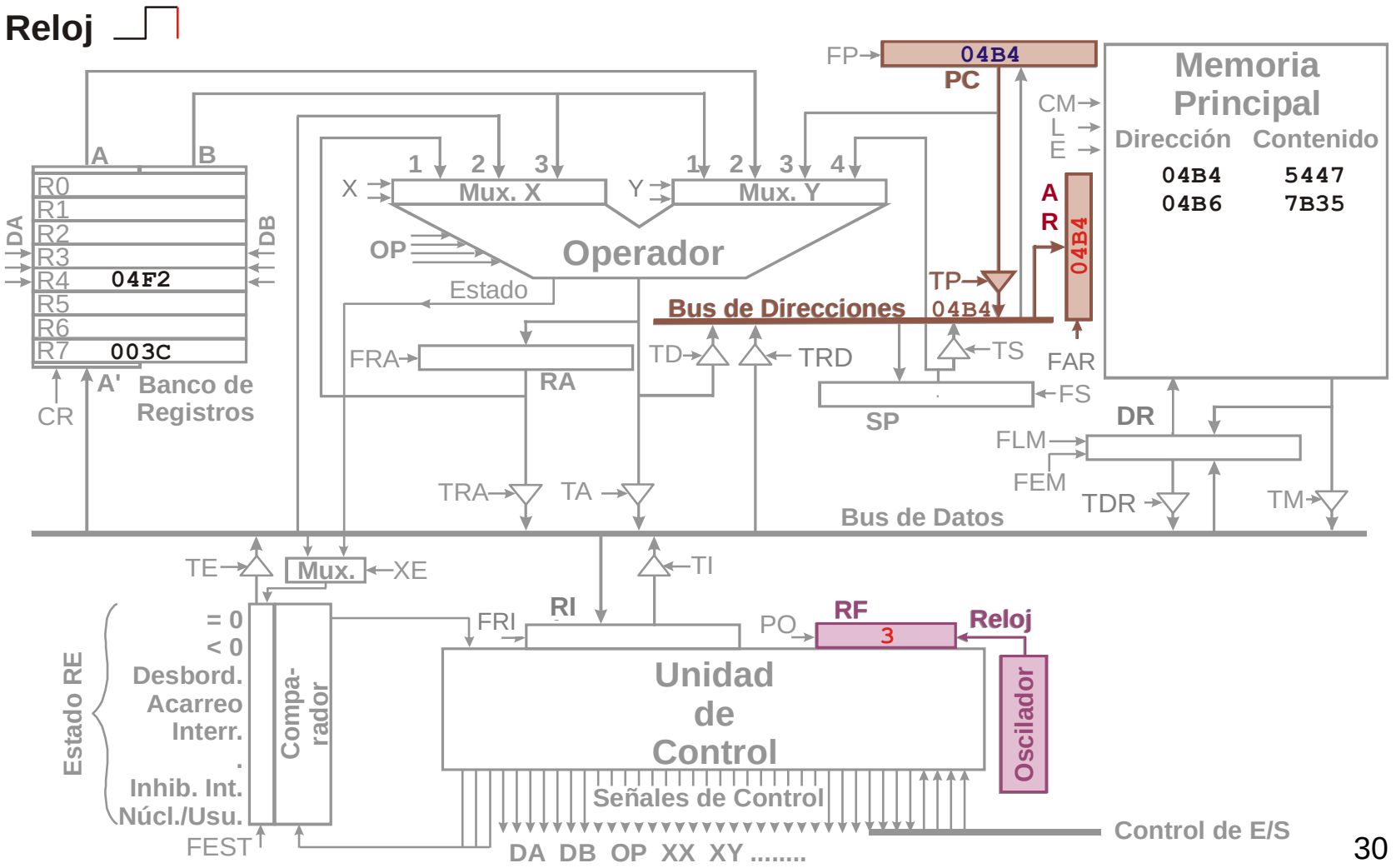
Se prepara: D ← PC



Prepara la lectura instrucción

Se realiza: D ← PC

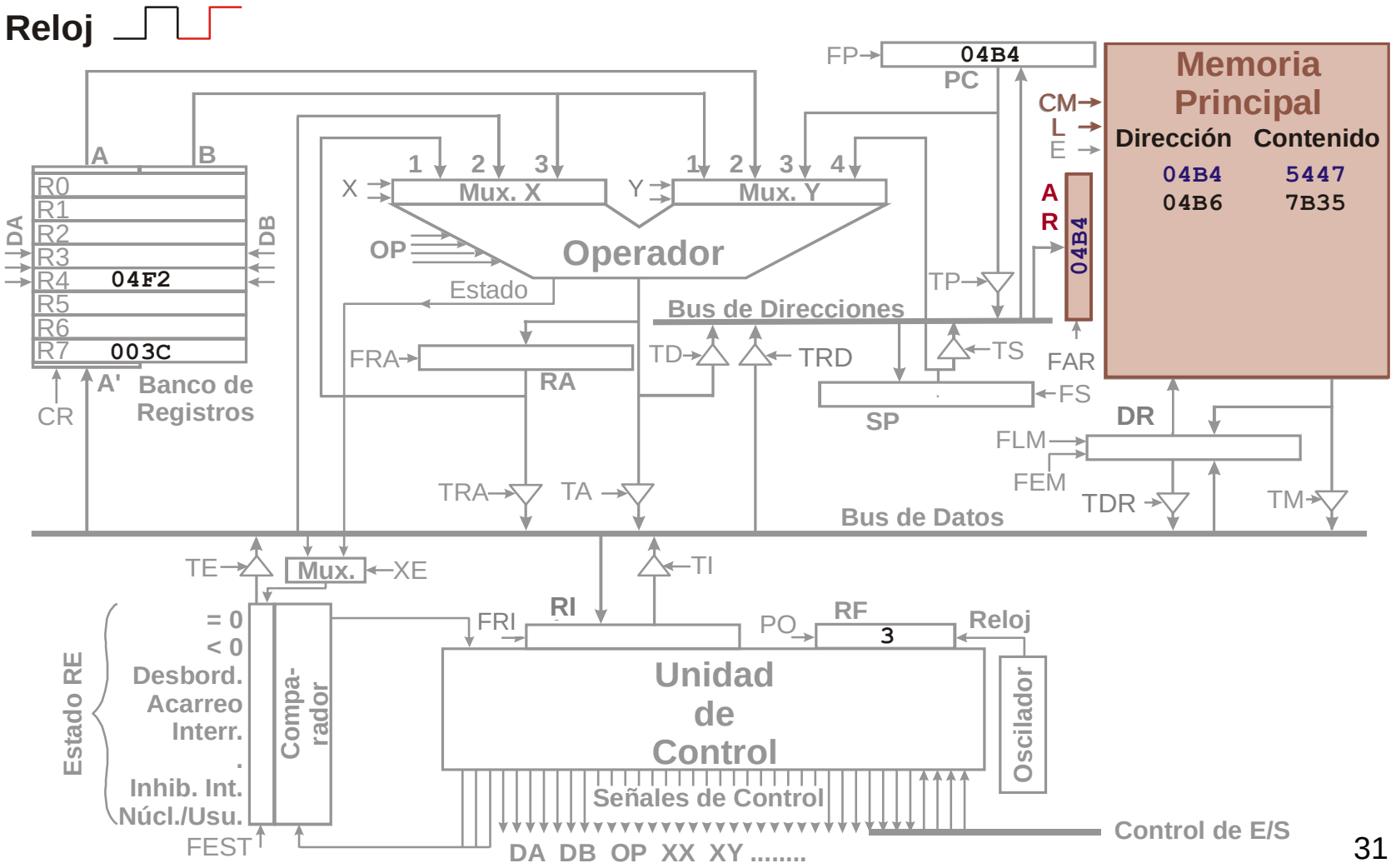
Se incrementa RF: RF ← RF + 1



ADD .R4,.R7

Lectura instrucción

Primer ciclo de memoria

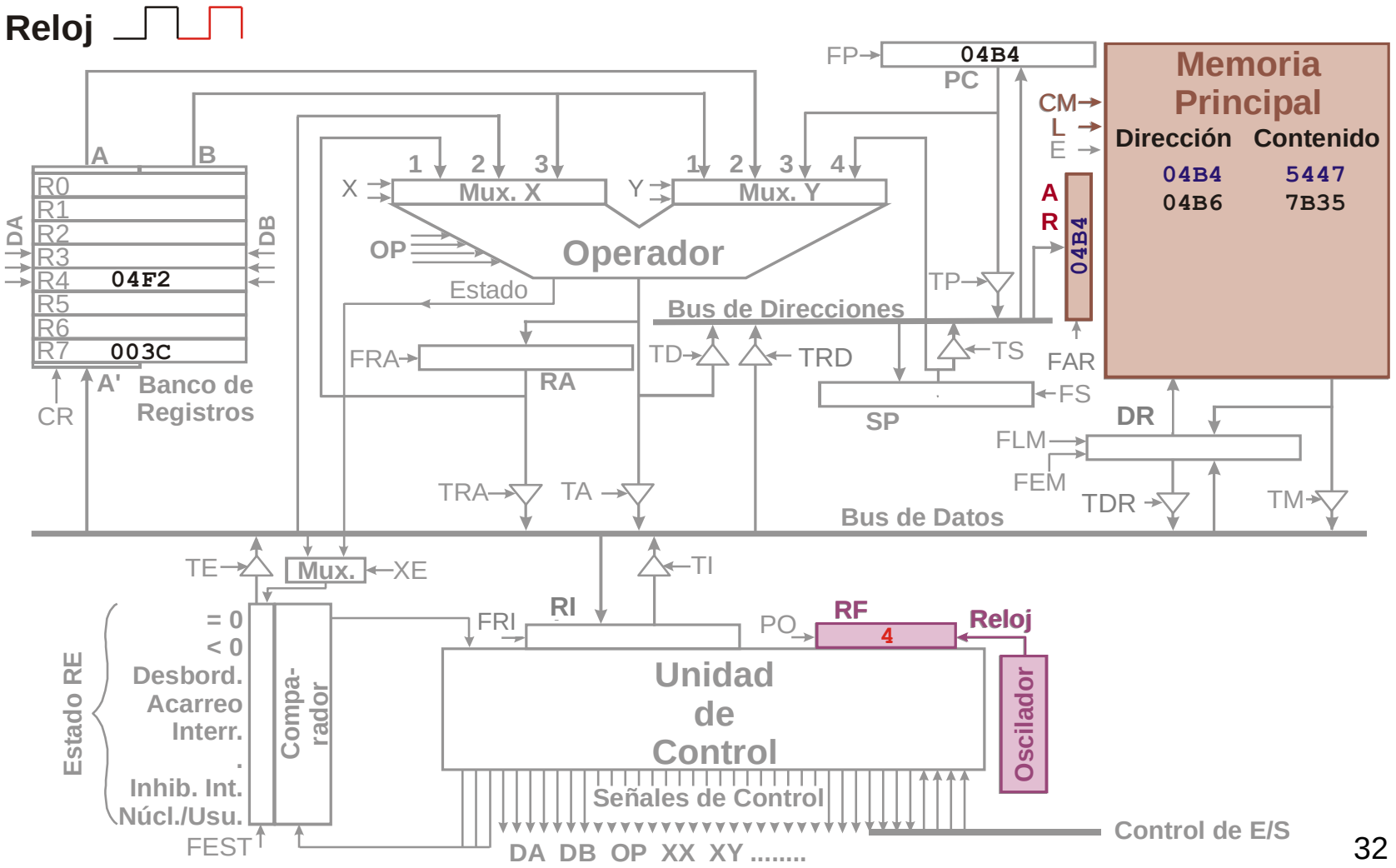


ADD .R4,.R7

Lectura instrucción

Primer ciclo de memoria

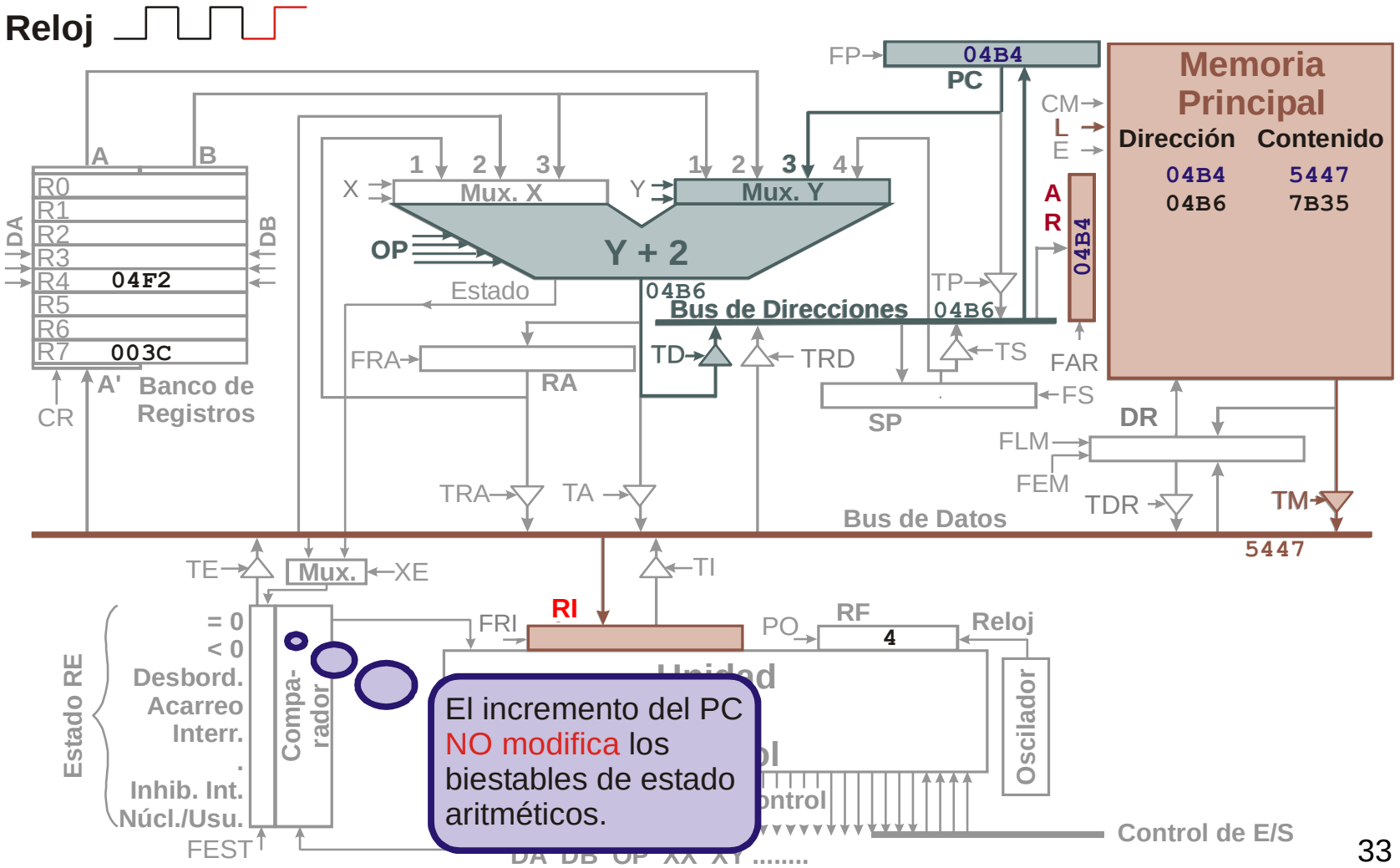
Se incrementa RF: RF ← RF + 1



Lectura instrucción

Segundo ciclo de memoria, se prepara: I ← M(D)

Se prepara incremento de PC: PC ← PC + 2



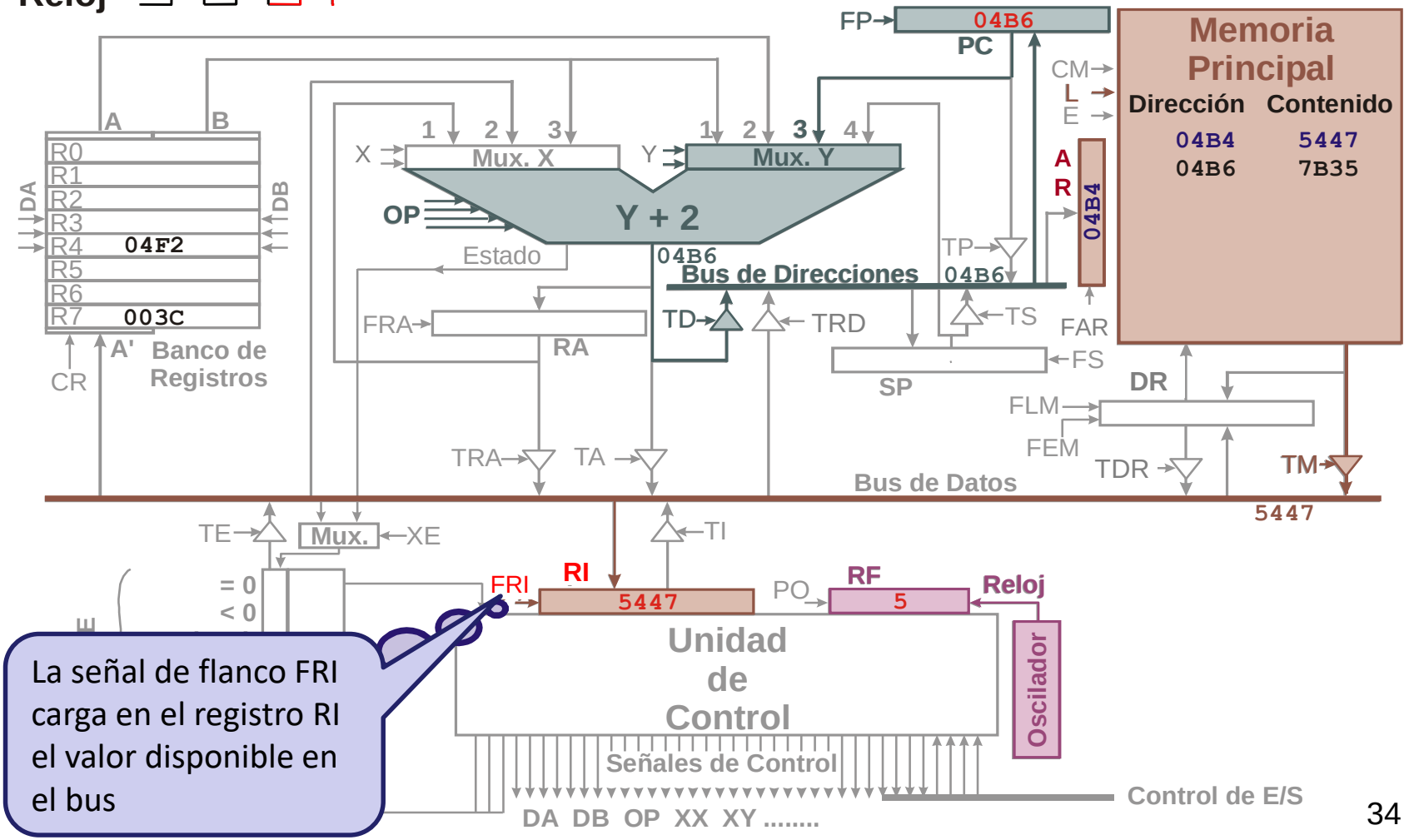
Lectura instrucción

Segundo ciclo de memoria, se realiza: I ← M(D)

Se realiza incremento de PC: PC ← PC + 2

Se incrementa RF: RF ← RF + 1

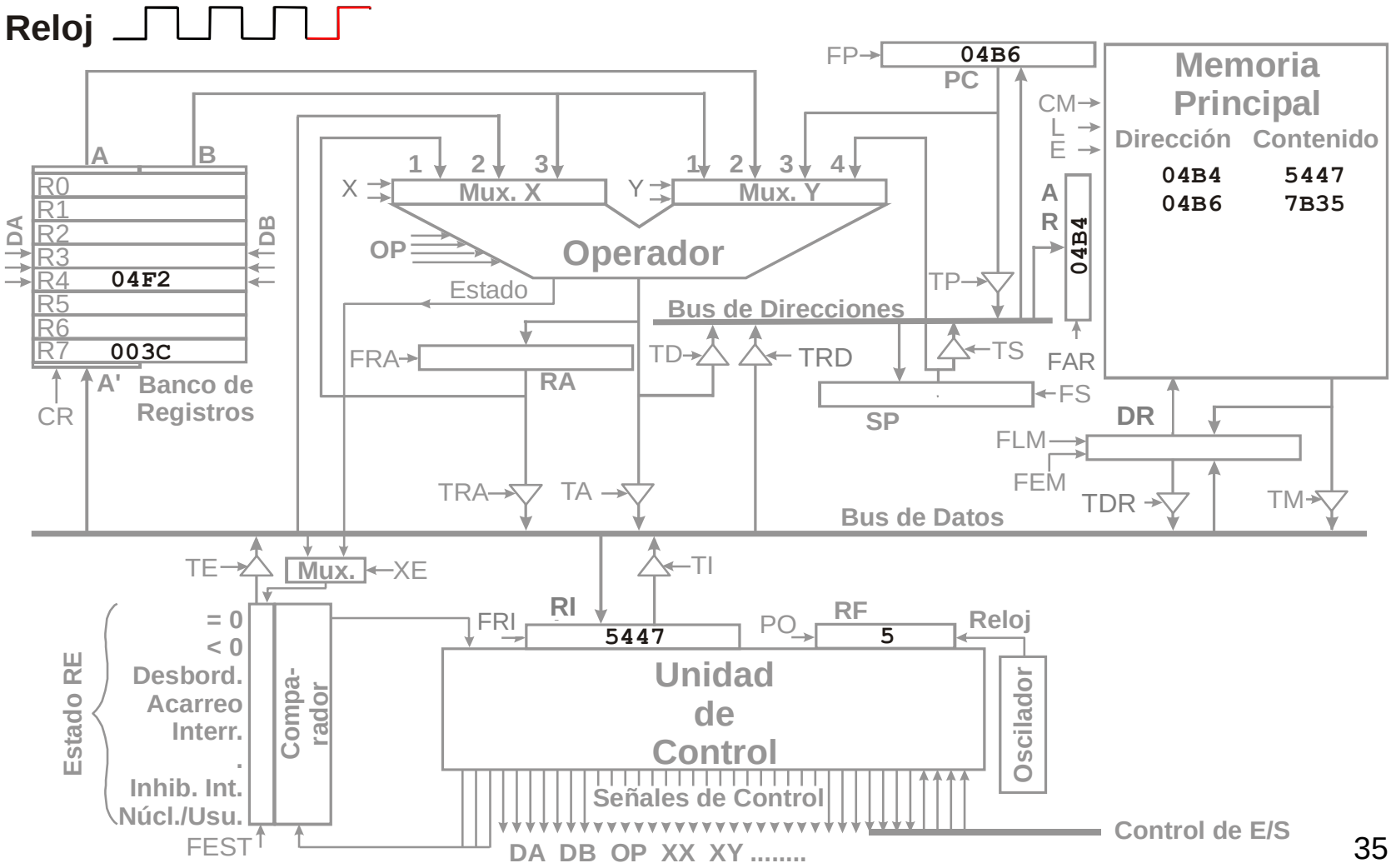
Reloj



La señal de flanco FRI carga en el registro RI el valor disponible en el bus

Decodificación instrucción

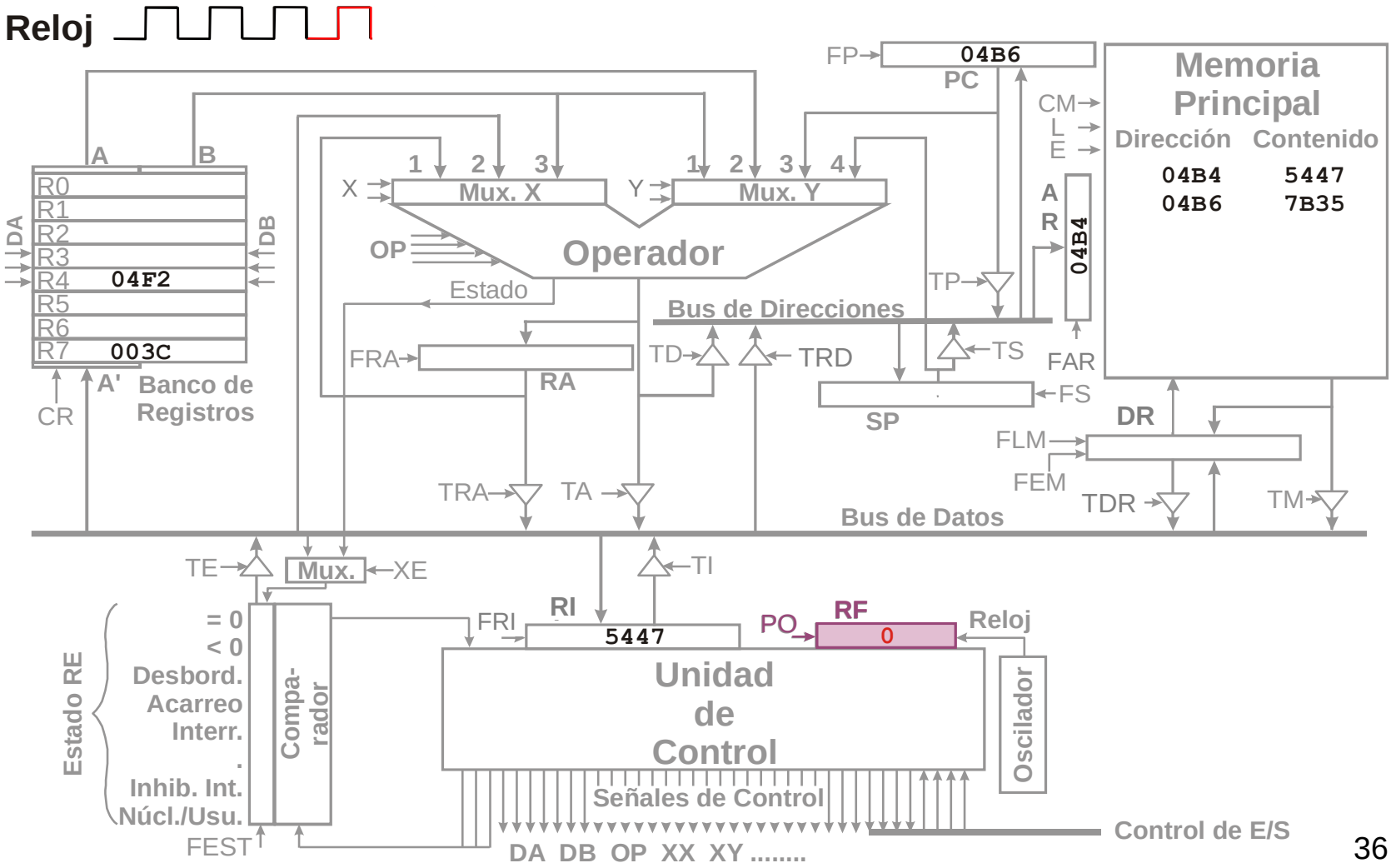
ADD .R4,.R7



ADD .R4,.R7

Decodificación instrucción

Se pone RF a 0: RF ← 0

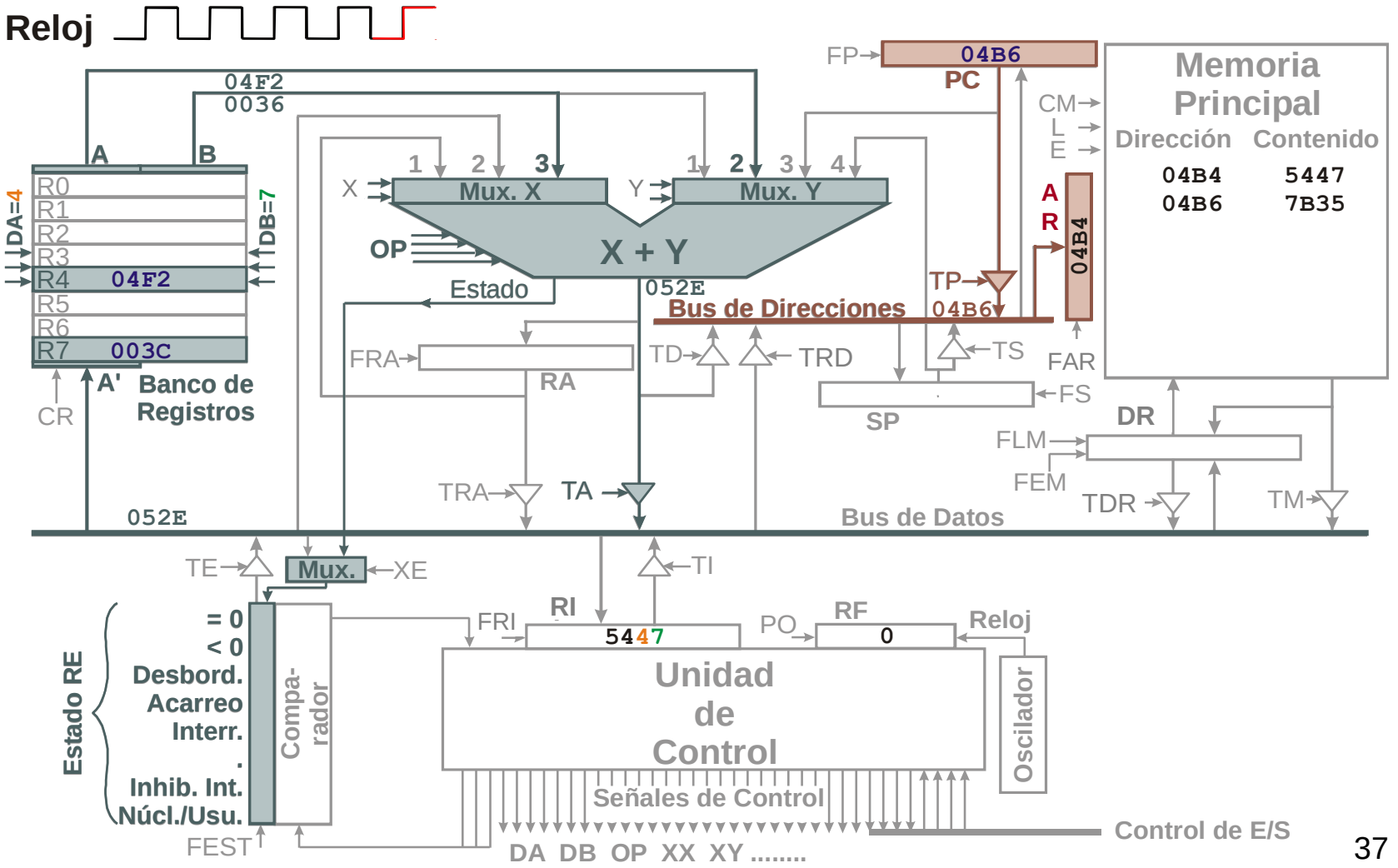


ADD .R4,.R7

Ejecuta instrucción y prepara lectura instrucción siguiente

Se prepara suma: R4 ← R4 + R7; RE ← Estado ALU

Se prepara: D ← PC



ADD .R4,.R7

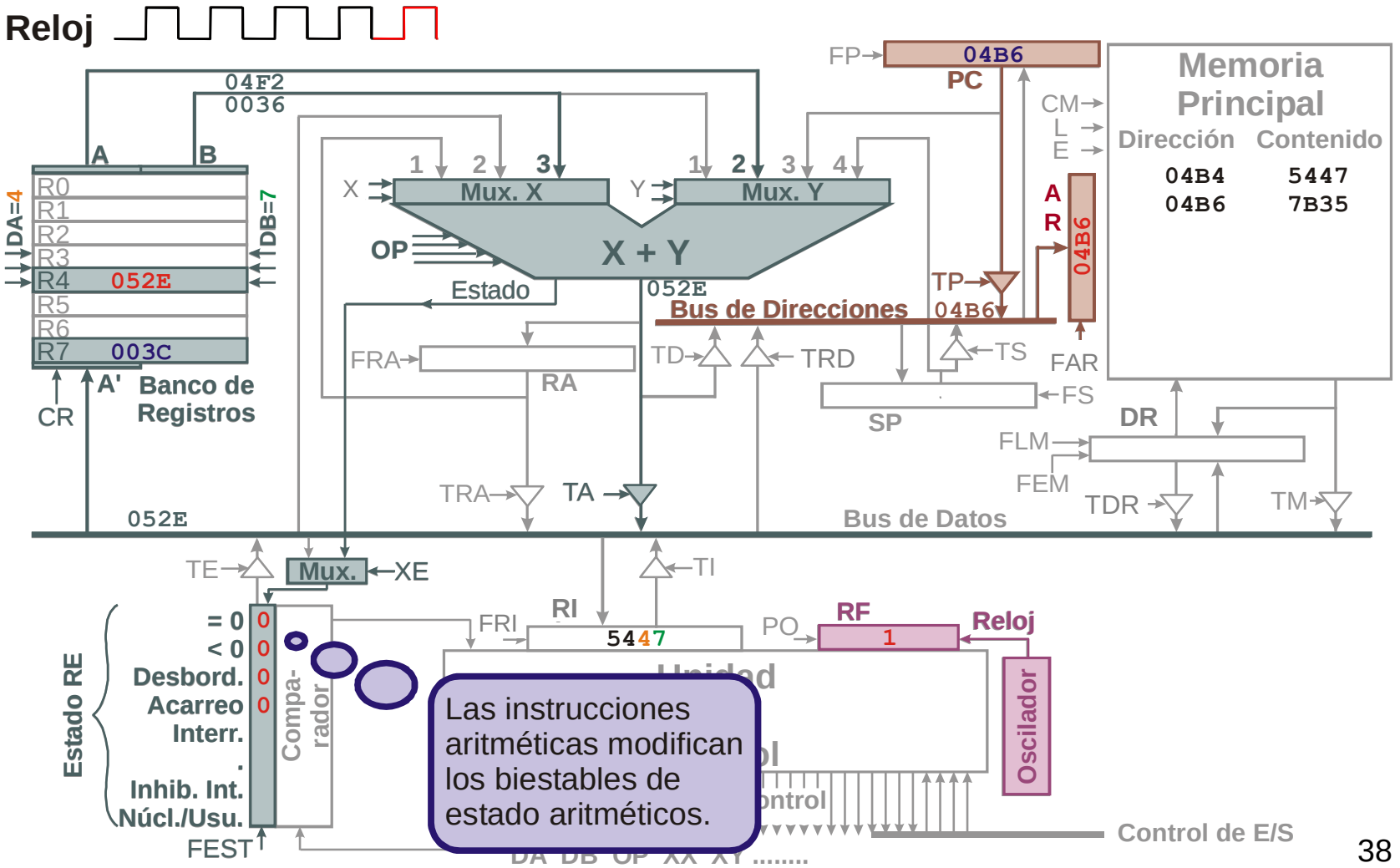
Ejecuta instrucción y prepara lectura instrucción siguiente

Se realiza suma: R4 ← R4 + R7; RE ← Estado ALU

Se realiza: D ← PC

Se incrementa RF: RF ← RF + 1

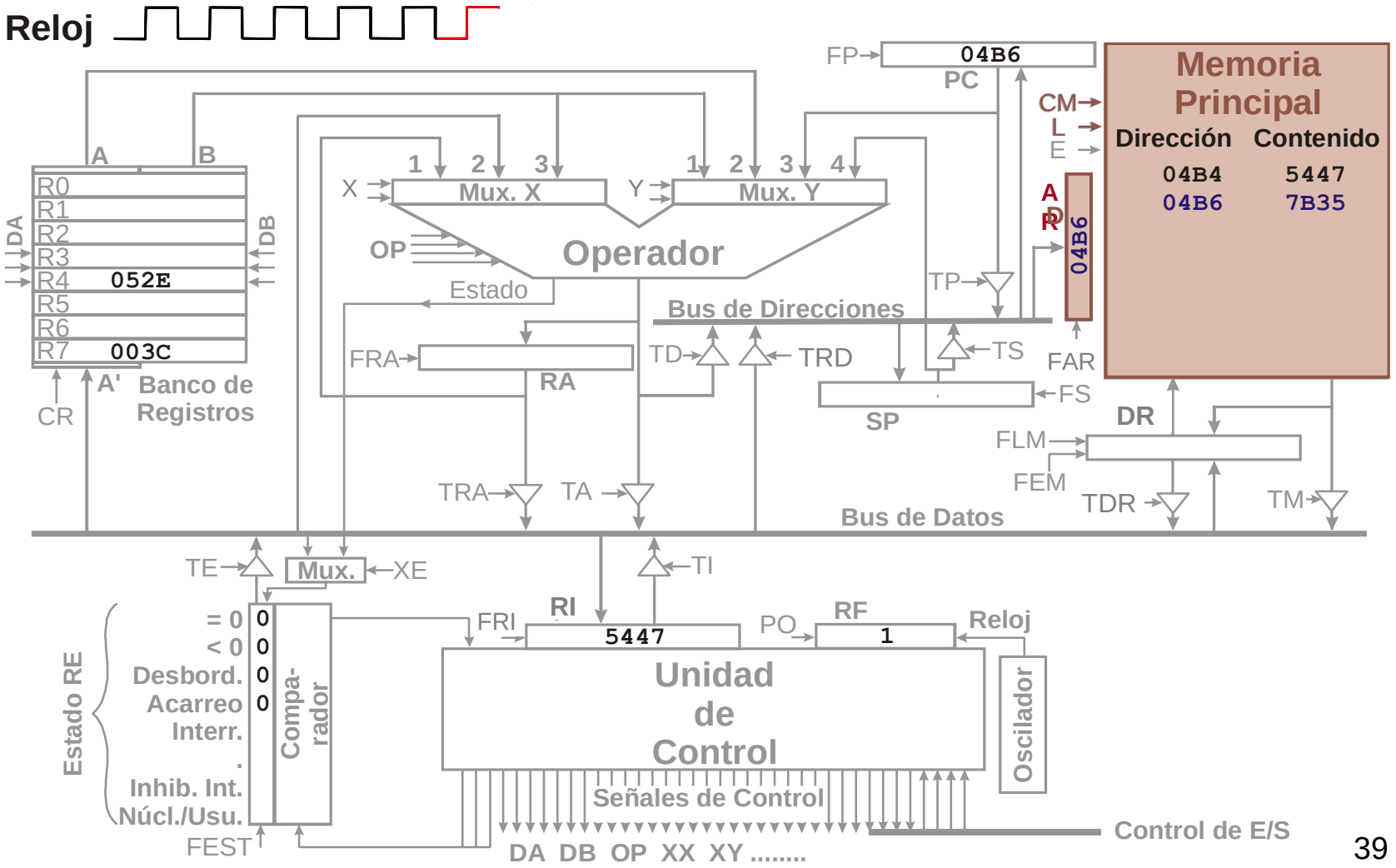
Reloj



Las instrucciones aritméticas modifican los biestables de estado aritméticos.

Lectura instrucción siguiente

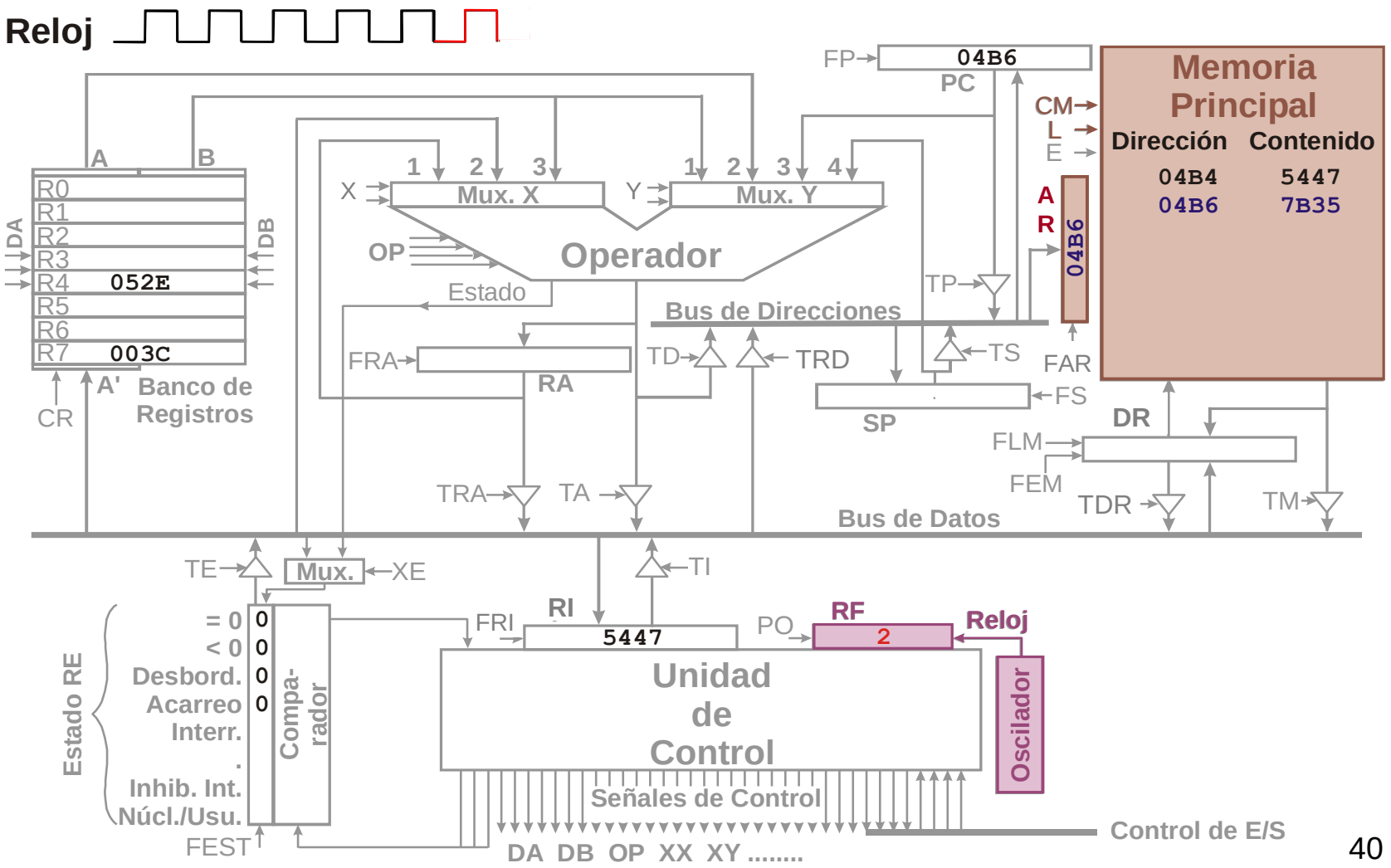
Primer ciclo de memoria



Lectura instrucción siguiente

Primer ciclo de memoria

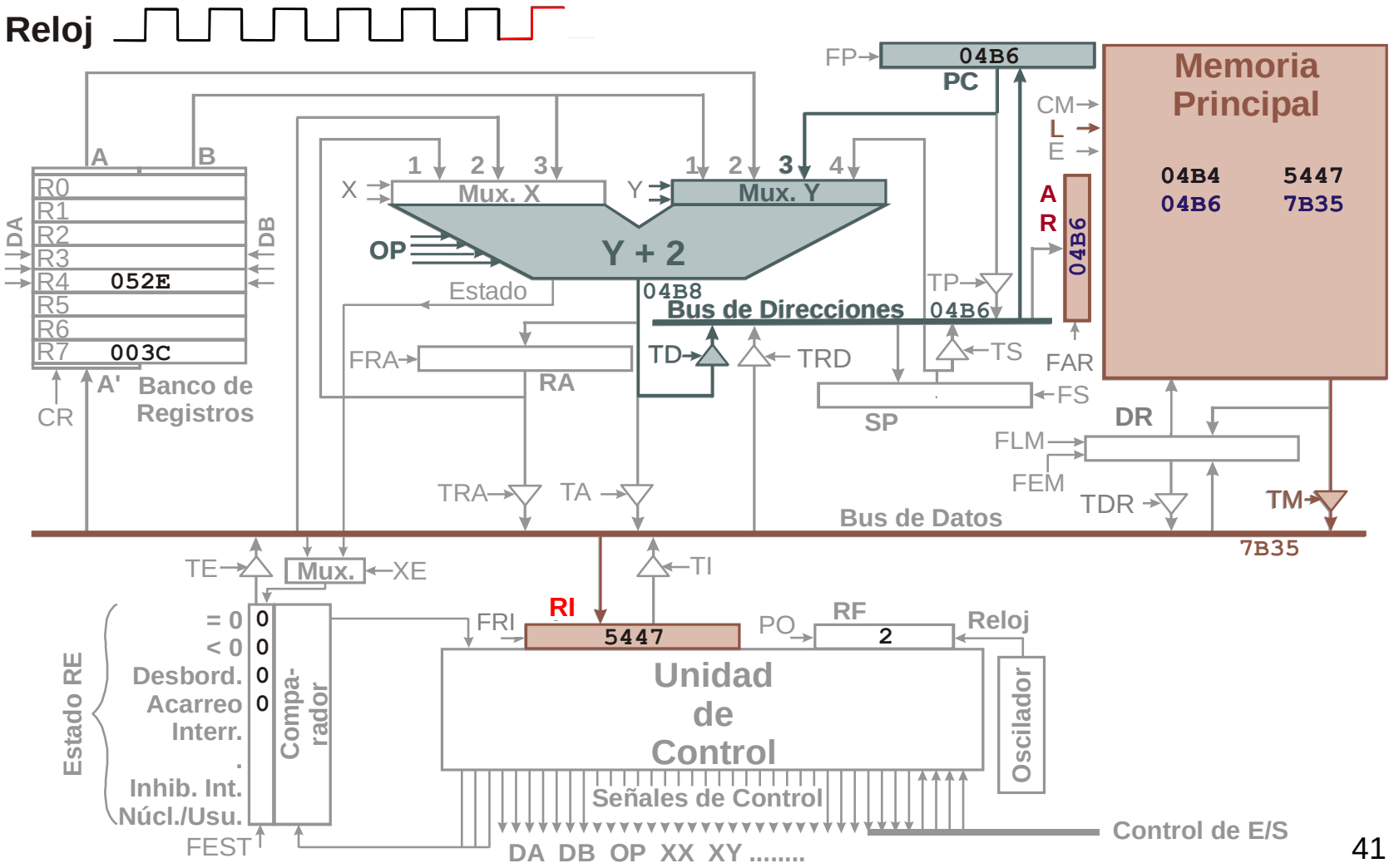
Se incrementa RF: RF ← RF + 1



Lectura instrucción

Segundo ciclo de memoria, se prepara: I ← M(D)

Se prepara incremento de PC: PC ← PC + 2

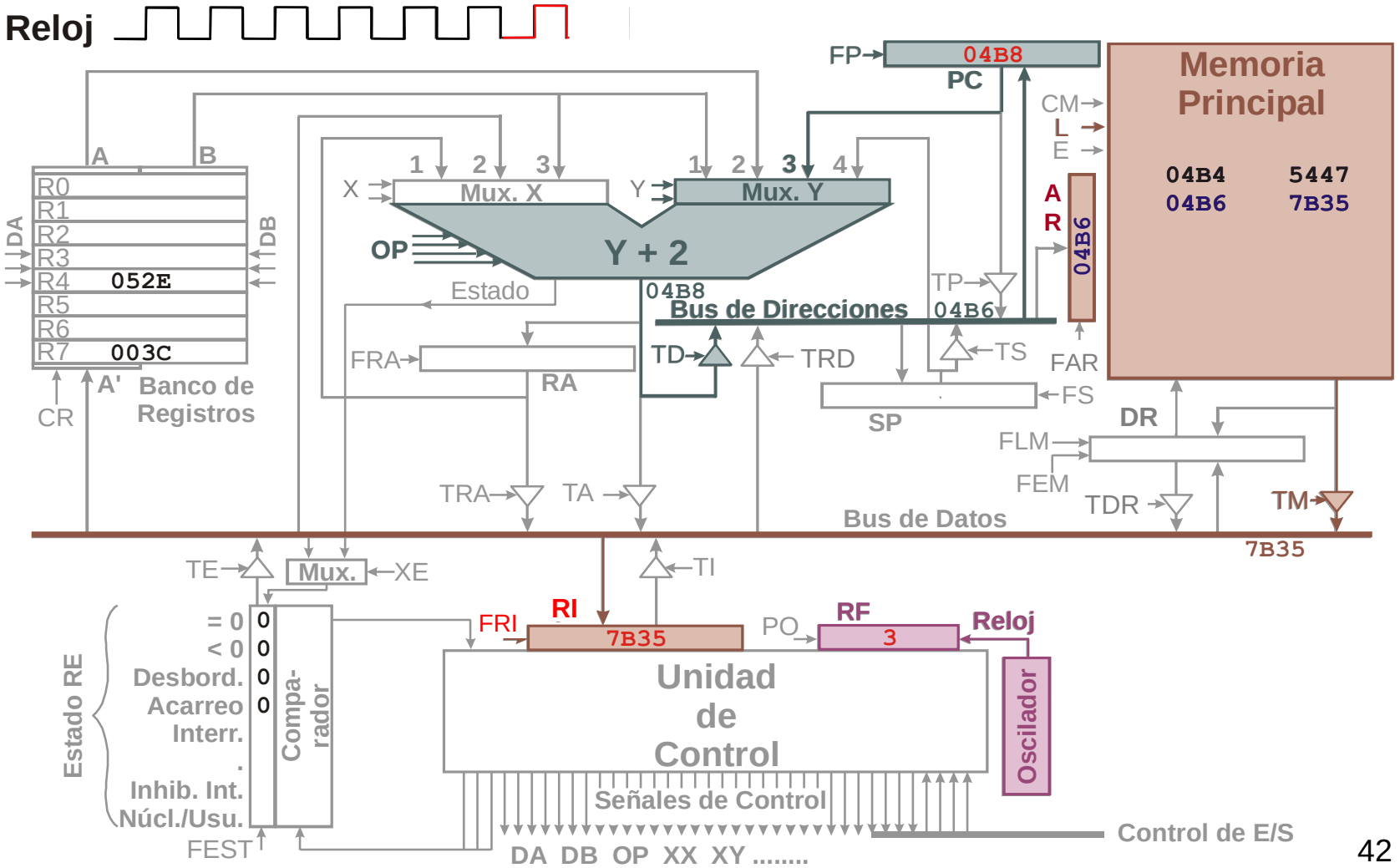


Lectura instrucción

Segundo ciclo de memoria, se realiza: I ← M(D)

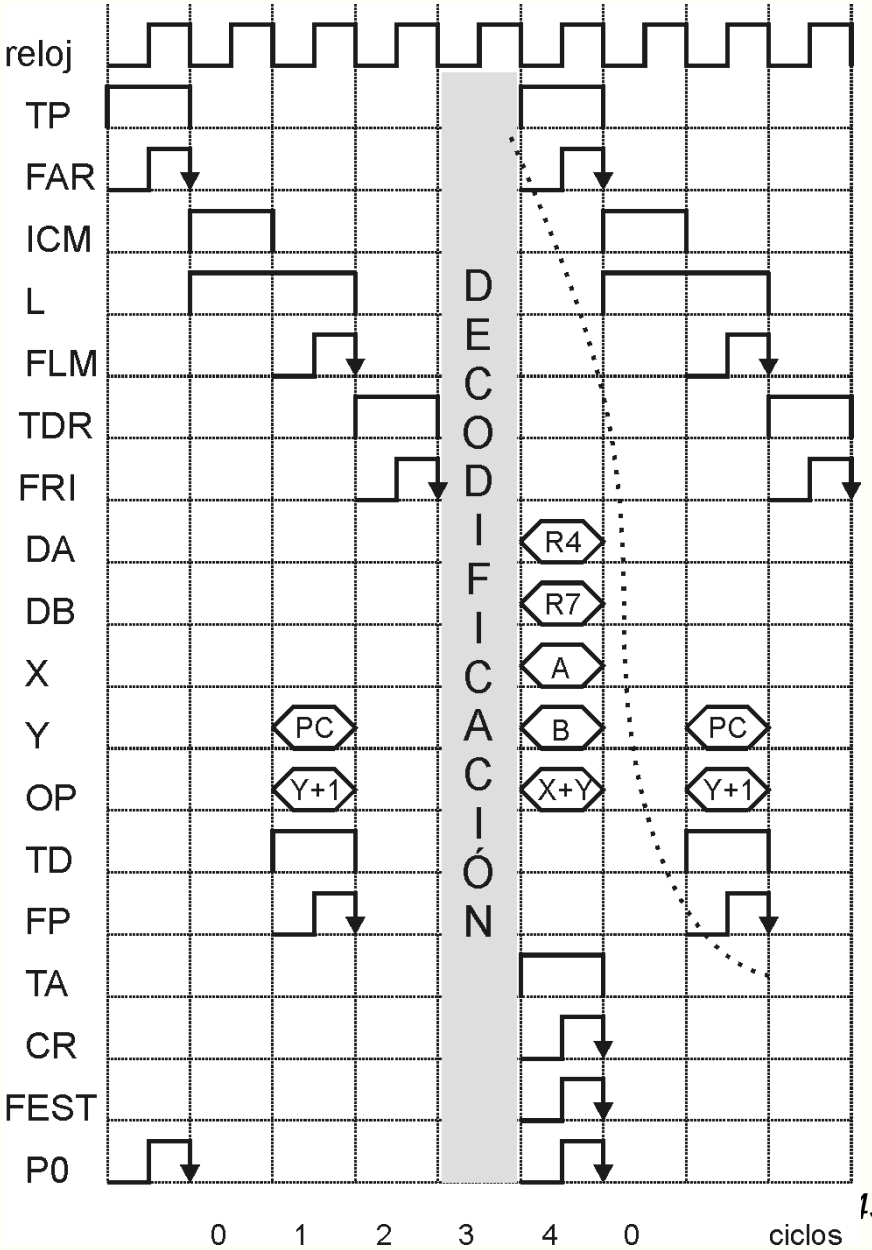
Se realiza incremento de PC: PC ← PC + 2

Se incrementa RF: RF ← RF + 1



Cronogramas

- *ADD .R4, .R7*
- *Lectura de la instrucción*
  - PC → AR (1 ciclo)
  - M(AR) → DR, PC + 1 → PC (2 ciclos)
  - DR → RI (1 ciclo)
- *Decodificación* (1 ciclo)
- *Ejecución y fetch sig. Instrucción*
  - R4+R7 → R4 y FEST;
    PC → AR (1 ciclo)
  - M(AR) → DR, PC + 1 → PC (2 ciclos)
  - DR → RI (1 ciclo)



Unidad de Control

- Indice
  - Introducción
  - Operaciones elementales
  - Estructura de un computador elemental y sus señales de control
  - Cronogramas
  - ▶ Diseño de la UC
  - UC microprogramada
  - Control de excepciones

Diseño de la UC

- El diseño de la UC exige haber definido previamente
  - El repertorio de instrucciones (formatos y direccionamientos)
    - A nivel de cronogramas o de operaciones elementales
  - La estructura del computador

    →Conjunto de cronogramas
- La UC funciona como un TRADUCTOR

  RI(CO, MD), Reloj, Estado, señ ext → señales de control

  Ej: CO: 8 bits, 1 instr=16 ciclos → RF: 5 bits, …

  traductor de unas 20-30 señales de entrada y 150 señales salida

Diseño de la UC

- Formas de diseño de la UC
  - UC CABLEADA
    - Utilizando exclusivamente puertas lógicas
  - UC MICROPROGRAMADA
    - Utilizando una memoria de control

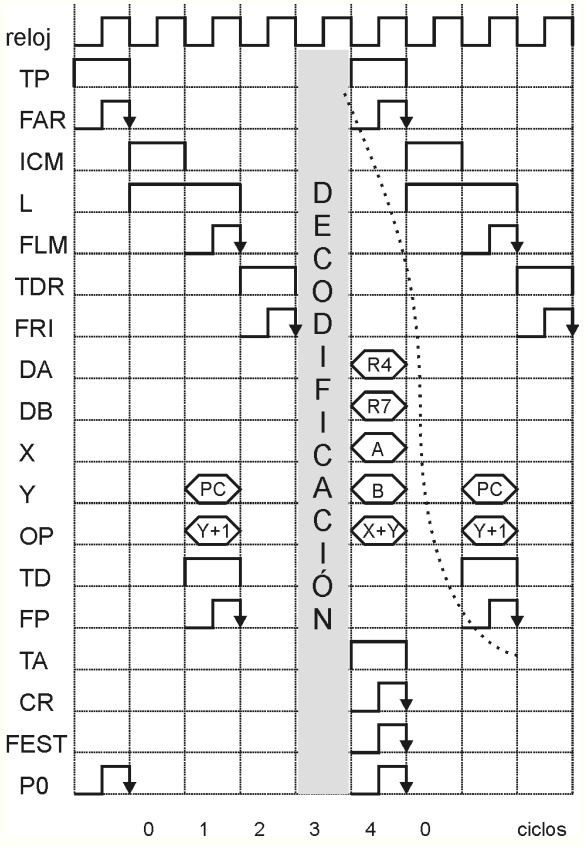
Diseño de la UC

- UC CABLEADA
  - Se construye mediante métodos generales del **diseño lógico** (circuito secuencial o combinacional)
  - Más **rápidas**
  - Diseño y construcción **complejo y costoso**
  - Las técnicas actuales (CAD, comp silicio) permiten que sea solución viable y válida cara a construir **máquinas rápidas**

Diseño de la UC

- UC MICROPROGRAMADA
  - Utiliza una **memoria** para almacenar el estado de las **señales de control** en cada periodo de la ejecución de cada instrucción.
  - Generar el **cronograma** de cada instrucción (ejecutar la Inst) es ir leyendo de esta memoria (Memoria de Control, **MC**).

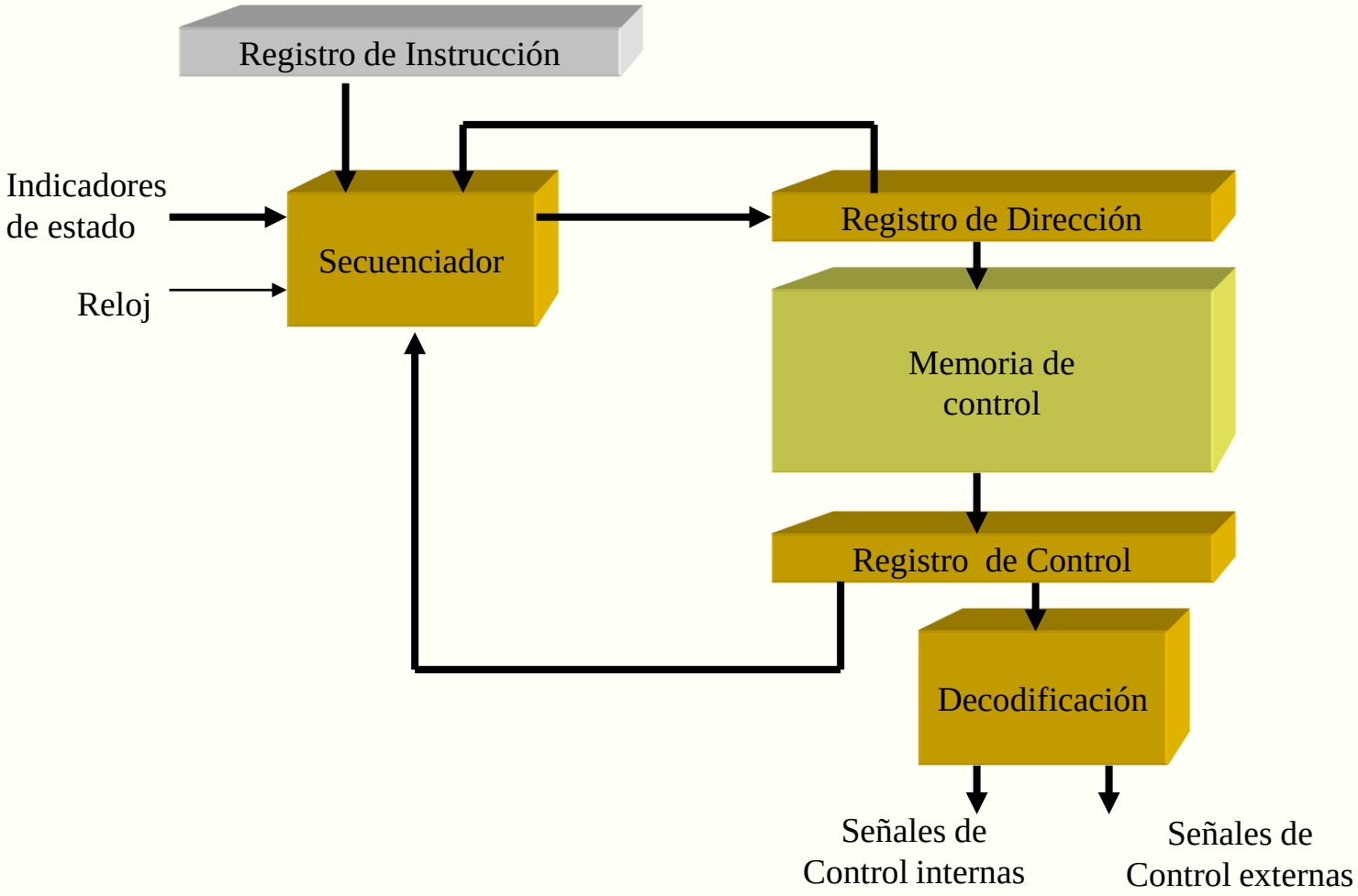
Diseño de la UC



Memoria de Control

TP	FAR	ICM	L	FLM	TDR	DA				DB				FRI	X		Y		OP				TD	FP	TA	CR	FEST	P0	...	...
x	x	x	x	x	x	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0	1	0
1	0	0	0	1	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	1	0	1	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0
1	0	0	0	1	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
1	0	1	0	1	1	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
x	x	x	x	x	x	0	0	1	1	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	1	1	0	0	0
x	x	x	x	x	x	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
x	x	x	x	x	x	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0
x	x	x	x	x	x	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
x	x	x	x	x	x	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x	x	x	x	x	x	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
x	x	x	x	x	x	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Diseño de la UC



Diseño de la UC

UC CABLEADA	UC MICROPROGRAMADA
Ventajas ●Alta velocidad de funcionamiento ●Implementaciones más pequeñas (reducido número de componentes)	Ventajas ●Son sistemáticas con un formato bien definido ●Pueden ser fácilmente modificables durante el proceso de diseño
Inconvenientes ●Difícil realizar modificaciones en el diseño: modificación → rediseño ●No tienen estructura común	Inconvenientes ●Requieren más componentes para su implementación ●Tienden a ser más lentas que las unidades de control cableadas debido a que tienen que realizar operaciones de lectura de una memoria para obtener las señales de control
▪Técnica de diseño favorita en las arquitecturas RISC	▪Emulación ●Se puede utilizar un microprograma para que la UC interprete otro lenguaje máquina distinto (una máquina a emular) sin necesidad de realizar modificaciones en el hardware de la unidad de control -> cambiando sólo el microprograma!

Diseño de la UC

- Conceptos básicos
  - UC MICROPROGRAMADA
  - MICROINSTRUCCIÓN
  - MICROPROGRAMA

Conceptos básicos

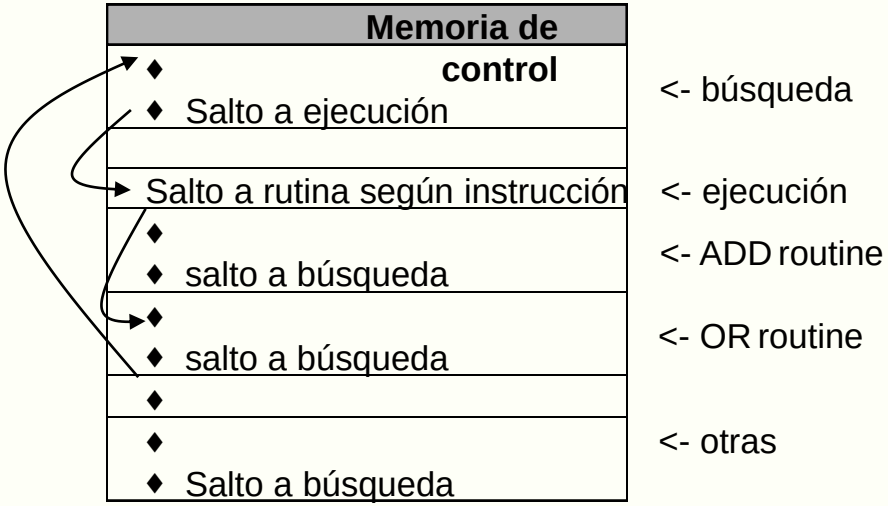
- **Microinstrucción**
  - **Cadena de ceros y unos** que
    - representa la activación o no del conjunto de señales de control durante un ciclo de reloj
    - y la posibilidad de decidir condicionalmente qué microinstrucción se debe ejecutar a continuación
  - **Cada palabra** de la memoria de control es una **microinstrucción**.

Conceptos básicos

Microprograma

- Secuencia de μI´s cuya ejecución permite interpretar una instrucción
- Ejecutar una microinstrucción ≈ Leer una μI de la MC
- Una vez rellenada la memoria de μI´s:

  el control es tarea de (micro)programación
- Cambiando microprogramas se puede ejecutar otros repertorios de instrucc (simular otras arquit) **-> EMULACIÓN!**



Conceptos básicos

- **Memoria de Control**
  - Memoria que almacena los **microprogramas** o (conjuntos de μI´s) a partir de los que se obtienen las señales de control necesarias para la ejecución del repertorio de instrucciones
  - Suelen ser ROM

Diseño de la UC

UC MICROPROGRAMADA

- Condiciones básicas:
  1. Capacidad suficiente de MC
  2. Procedimiento de correspondencia

     Inst → dir MC comienzo su μprograma
  3. Secuenciamiento de μinstrucciones

Conceptos básicos

- **Formato de microinstrucciones**
  - **Señales de control**: Señales para el camino de datos
  - **Condiciones**: Bits para seleccionar la condición que se desea utilizar para en función de si es cierta o no ejecutar la siguiente microinstrucción o saltar a otra
  - **Siguiente μinstrucción**: Campo que indica la siguiente μinstrucción a ejecutar

Señales de control	Tipo Sec.	Cond	Siguiente μinstr

Diseño de la UC

Diseño del formato de microinstrucción

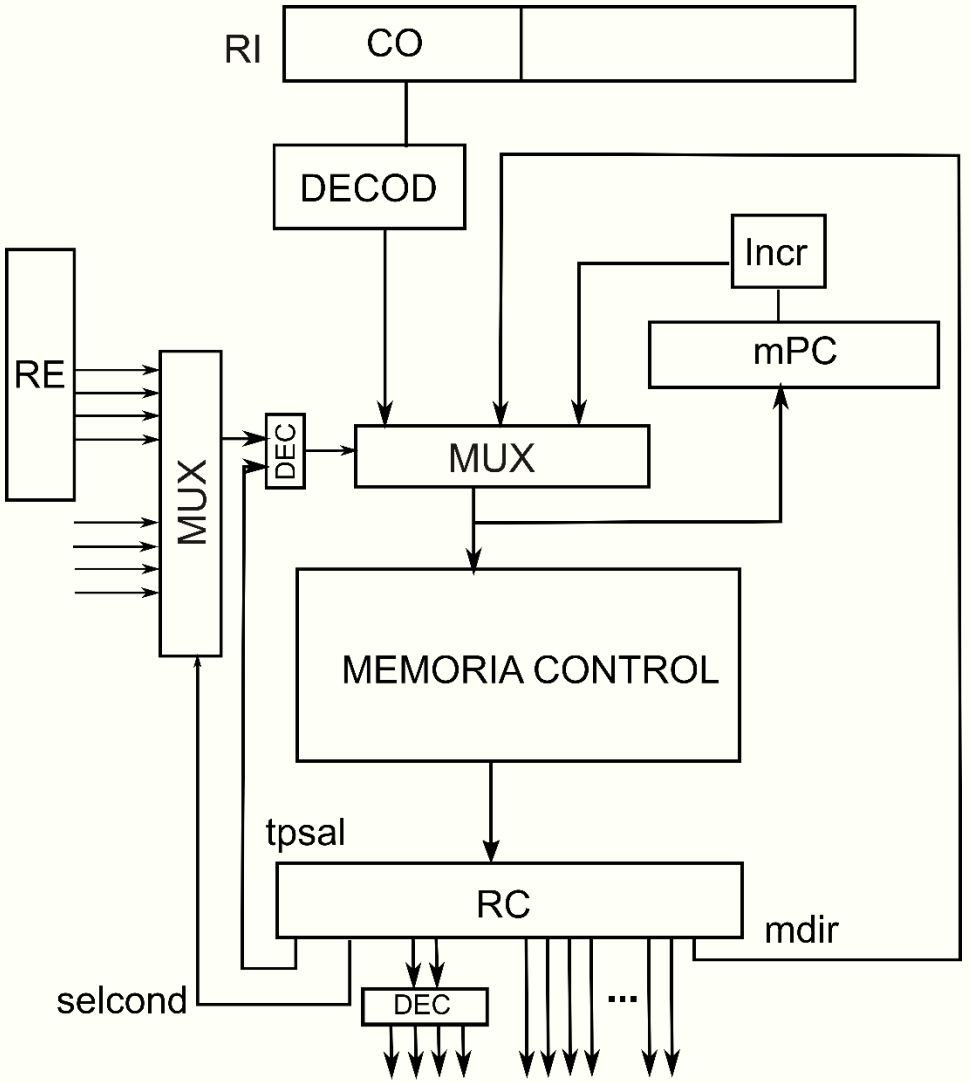
- Comenzar con la lista completa de señales de control
- Agrupar las señales de control con función similar en un mismo **CAMPO**
  - Controlar que una misma señal no esté en diferentes campos
- Campo de **CONTROL DE SECUENCIAMIENTO**: indica la siguiente mI a ejecutar
- Secuenciamiento: se deben almacenar en algún orden en MC, por ejemplo, secuencialmente (sec. implícito)
- Se puede aplicar CODIFICACIÓN:
  - Para reducir el tamaño de la mI
  - Para evitar que ciertas señales se activen al mismo tiempo (evitar errores)

    Ej: acceso a bus, operación de memoria, etc…

Diseño de la UC

Estructura básica de la UC mprogramada

- microsaltos condicionales



Unidad de Control

- Indice
  - Introducción
  - Operaciones elementales
  - Estructura de un computador elemental y sus señales de control
  - Cronogramas
  - Diseño de la UC
  - UC microprogramada
  - ► Control de excepciones

Control de excepciones

- Además de ejecutar la secuencia de instrucciones del programa de usuario, la UC debe controlar situaciones excepcionales
  - Qué ocurre si el CO no es válido?
  - Y si hay overflow?
  - Cómo y cuándo comunicar con algún periférico?
  - …

Control de excepciones

- Excepción [Overflow…]
  - Evento no planificado que para la ejecución de un programa. (interno)
    - Salvar el PC y Estado actual en pila
    - Pasar a modo SO (modo supervisor)
    - Saltar a rutina de SO que maneje excepciones internas
    - SO retornará después al programa, en su anterior estado
- Interrupción [E/S]
  - Un evento externo inesperado rompe la ejecución del programa

Excepciones

- Objetivos
  - *Detección de excepciones* – Cuándo y cómo verlas?
  - *Manejo de excepciones* – Qué hacer?

Control de excepciones

Excepciones

- Causadas por eventos internos
  - Condiciones de excepción (ej: overflow)
  - Errores (ej: error de alineamiento de memoria)
  - Fallos (ej: fallo de página)
  - Llamadas al sistema
- Síncronos a la ejecución del programa (se puede saber cuándo)
- Puede que el programa de usuario se retome tras tratar la excepción, o bien que se aborte

- TRAP: llamada al sistema

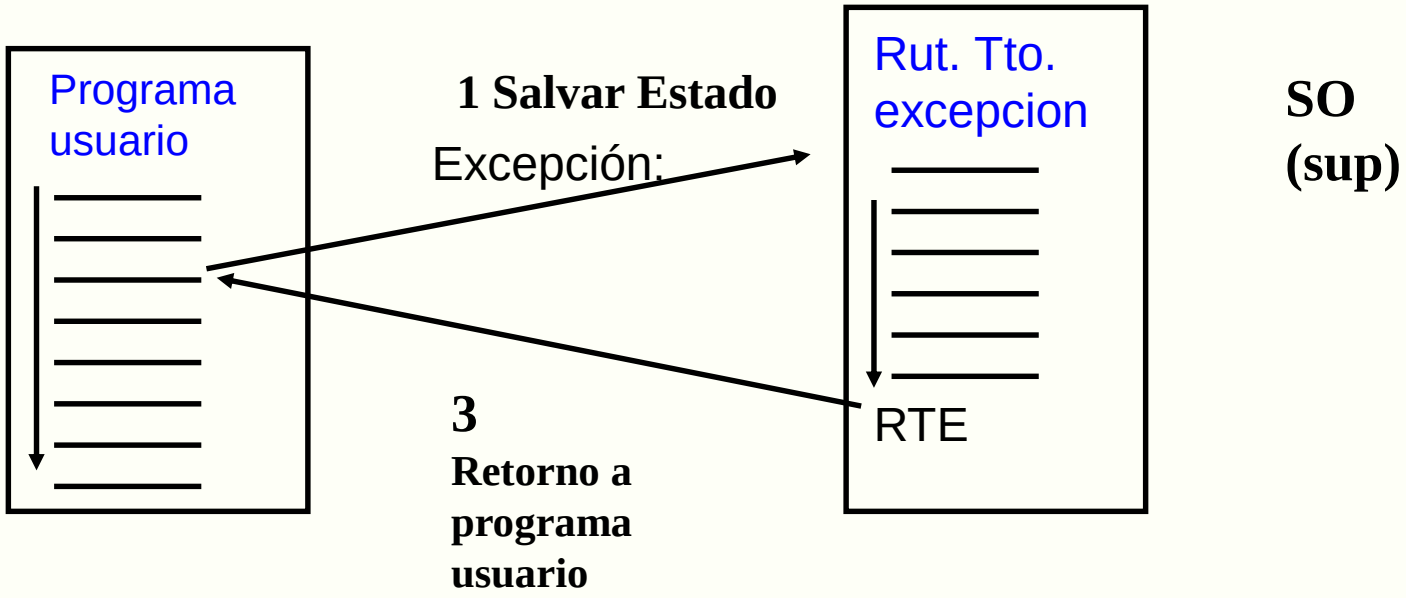
Proceso de excepciones

Interrupciones

- Causadas por eventos **externos** (peticiones de periféricos).
- Asíncronas a la ejecución del programa (cualquier instante)
- Se atienden entre Instrucciones (fetch)
- Mantienen sin ejecutarse un tiempo el programa de usuario

*FETCH**:* Si INT e INT_no_inhibidas ir a *Microrrut_proceso_excepcion*
Si no ir a FETCH

Control de excepciones



Manejo de excepciones

- Excepción = transferencia de control no programada

  El sistema maneja la excepción:
  - Guarda la dir de la Instrucción que causó la excepción
  - Salvaguarda el estado del programa de usuario
  - Trata la excepción
  - Retorna el control a usuario (salvo que se aborte)