



Estructura de Computadores

Tema 3: Procesador

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

1



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, UC**)
 - entender esta ejecución de instrucciones mediante su representación en el tiempo por **cronogramas**
- Comprender el **diseño** de la UC
 - ser capaz de descomponer 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

2



Bibliografía

- de Miguel, P. *"Fundamentos de los computadores"*, Paraninfo, 2004. 9ª Edición
- Patterson, D. A.; Hennessy, J. L. *Estructura y diseño de Computadores*. Ed. Reverté 2011. 4ª Edición
- Stallings, W. *"Organización y arquitectura de computadores"*, Prentice Hall, 2006, 7ª Edición

3



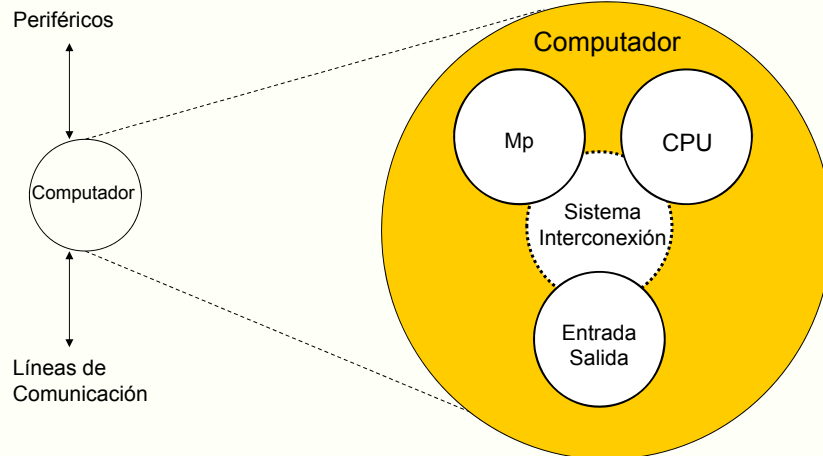
Introducción

- Estudiaremos:
 - **Unidad de control:**
 - Encargada de interpretar las instrucciones del programa y gobernar la ejecución de las mismas
 - **Camino de datos (*datapath*):**
 - Parte de la CPU donde se transfieren los datos procedentes de la memoria o registros internos, para obtener los resultados
 - Debe soportar el conjunto de operaciones que precisan las instrucciones del repertorio capaz de interpretar la unidad de control
- Organización de procesadores: ha evolucionado
 - desarrollo tecnológico
 - la necesidad de obtener altas prestaciones

4

Introducción

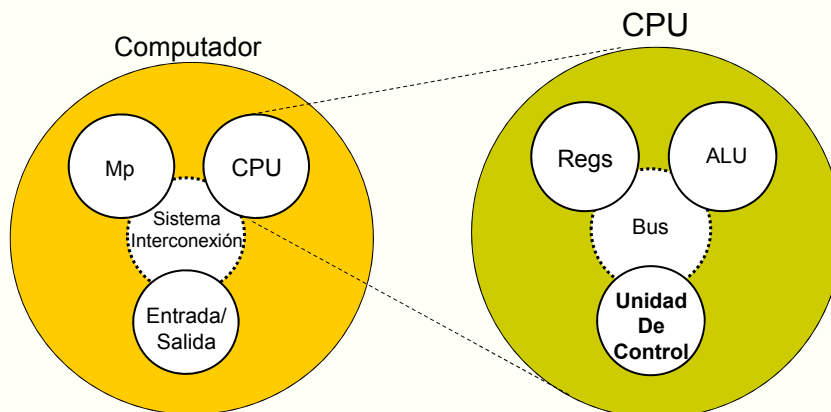
Visión global del computador y de la CPU



5

Introducción

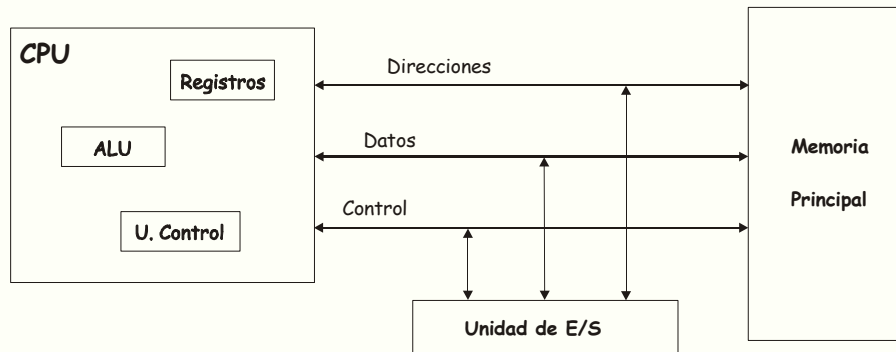
Visión global del computador y de la CPU



6



Esquema básico del computador Von Neumann. Componentes



7



Introducción

- Función básica de CPU:
ejecutar instrucciones
 1. Leer la instrucción en memoria (*fetch*)
 2. Decodificar la instrucción
para determinar qué acciones se han de realizar
 3. Ejecución: como máximo
 - Búsqueda de operandos
 - cálculo de operando
 - leer operando en memoria
 - Operación
 - Almacenar resultado
 4. Preparar la siguiente instrucción (ir a paso 1)

8



Introducción

- Funciones de la CPU:
 1. Ejecuta instrucciones (función básica)
 - Lectura (*fetch*) y decodificación de las instrucciones
 - Generación de órdenes para la ejecución
 - Secuenciamiento de las instrucciones
(decidir cuál es la siguiente que se ha de ejecutar)
 2. Resuelve situaciones anómalas
(desbordamiento, operación no válida, violación de privilegio, etc)
 3. Controla la comunicación con periféricos

9



Introducción

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

10

Introducción

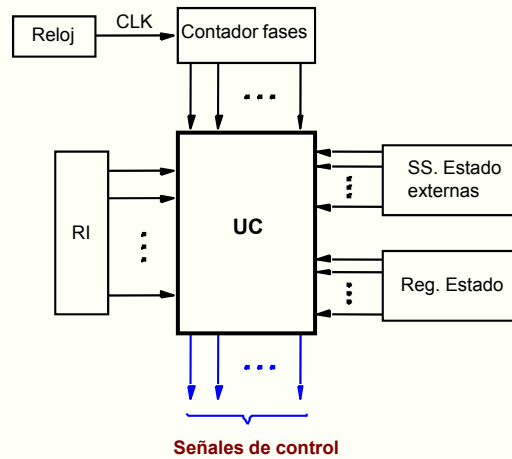
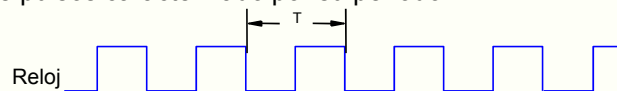


Figura Entradas y salidas de la UC.

11

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 $f = 100 \text{ MHz}$

$T = 10 \text{ ns}$ y conviene que una operación aritmética se realice en un tiempo algo menor que 10 ns
¿Y Mem?

12



Introducción

Ciclo de lectura (en Mem)

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)

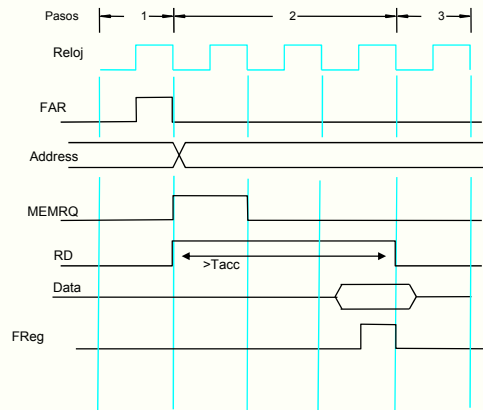


Figura . Temporización de la lectura en memoria

13



Introducción

Ciclo de escritura (en Mem)

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

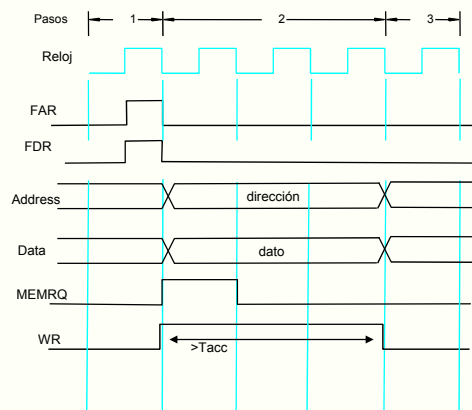


Figura . Temporización de la escritura en memoria

14



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

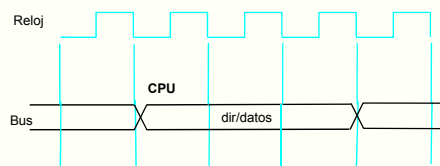


Figura . Ciclo de bus

15



Introducción

Evaluación del rendimiento

- Supóngase los tiempos de ejecución:
 - Acceso a memoria: 8 ns
 - ALU y sumadores: 2 ns
 - Acceso a registros: 1 ns
- ¿Cuál 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?

16



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

17



Operaciones Elementales

- **Ciclo de instrucción:** conjunto de acciones que requiere la ejecución de una instrucción
 1. Leer la instrucción en memoria
 2. Decodificar la instrucción
 3. Ejecución: como máximo
 - Búsqueda de operandos, que puede conllevar
 - Cálculo de operando
 - Leer operando en memoria
 - Operación
 - Almacenar resultado
 4. Preparar la siguiente instrucción (ir a paso 1)

18



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**

19



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 preestablecido
 - Conflictos en los elementos Hw

20



Operaciones Elementales

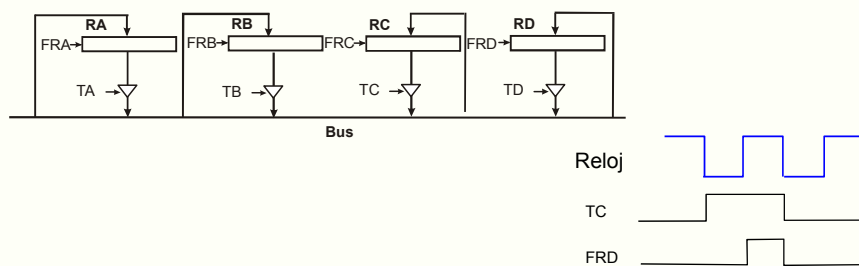
- Tipos:
 - **De transferencia**
Llevar información de un origen a un destino
 - **De proceso**
Llevar información de un origen a un destino, pero esta pasa por un operador

21



Operaciones Elementales

- **OE de transferencia**
 - Llevar información de un ORIGEN a un DESTINO
 1. Establecer camino físico entre salida de origen y entrada de destino
 2. Activar la señal del destino para que tome la informaciónEj: Transferencia a través de un bus de dos registros **C** → **D**



22



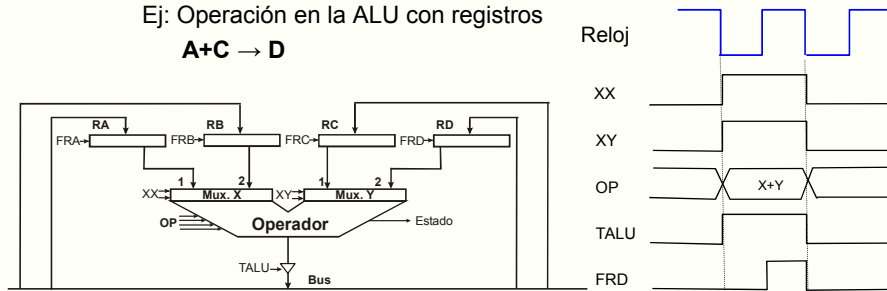
Operaciones Elementales

■ 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

$$A + C \rightarrow D$$



23



Operaciones Elementales

■ Reglas de agrupación de OEs

- Respetar el **orden lógico** entre las 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; memoria no puede leer y escribir al mismo tiempo, etc.*
- La información debe guardarse en algún sitio: ORIGEN y DESTINO deben poder almacenar información (reg o mem)

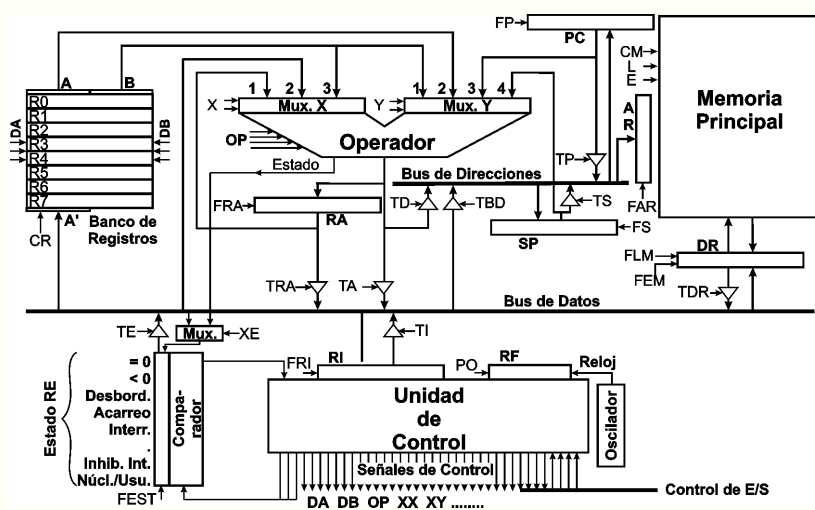
24

Unidad de Control

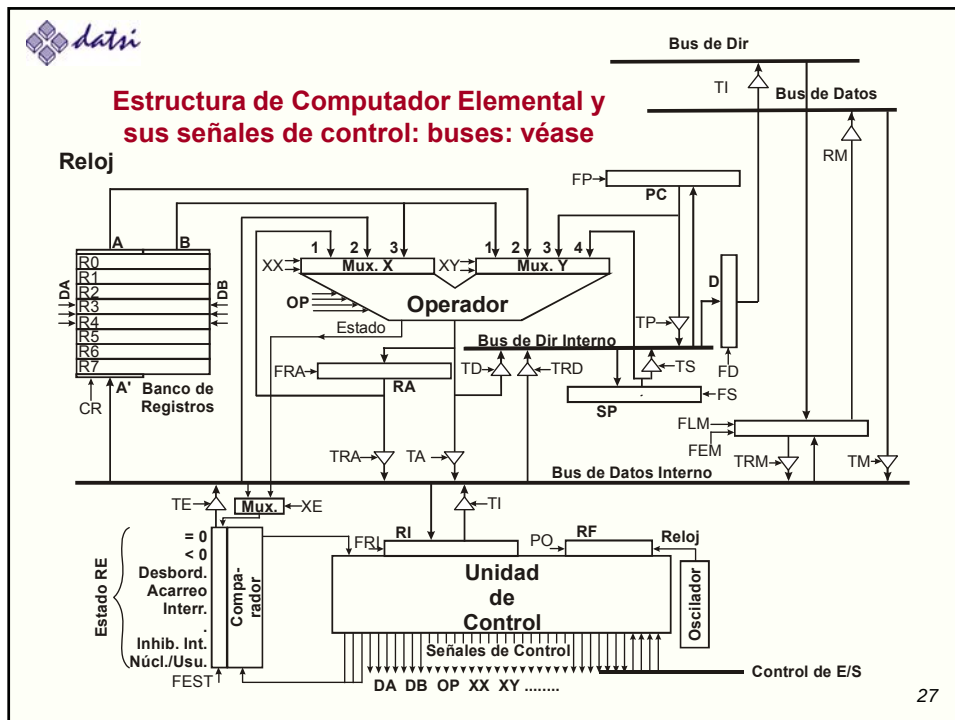
- Índice
 - Introducción
 - Operaciones elementales
 - Estructura de un computador elemental y sus señales de control
 - Cronogramas
 - Diseño de la UC
 - UC microprogramada
 - Control de excepciones

25

Estructura de Computador Elemental y sus señales de control



26

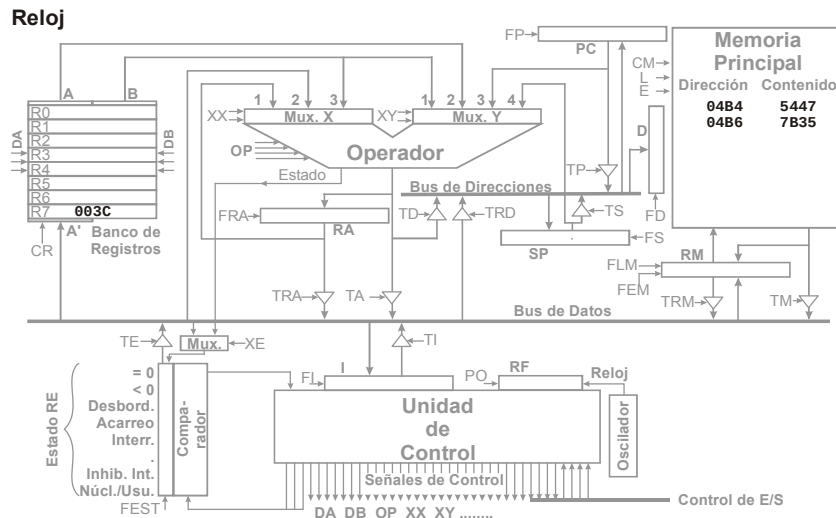


 **Unidad de Control**

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

28

ADD R4, R7



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:

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

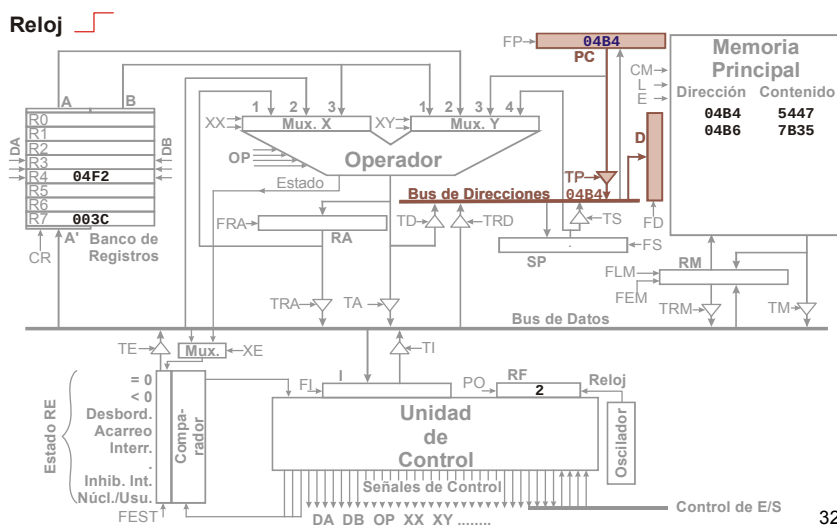
Además de estas operaciones, se incrementa el contador de programa, para ir preparando la lectura de la siguiente instrucción.

$PC \leftarrow PC + 2$

Prepara la lectura instrucción

Se prepara: $D \leftarrow PC$

ADD R4, R7



32

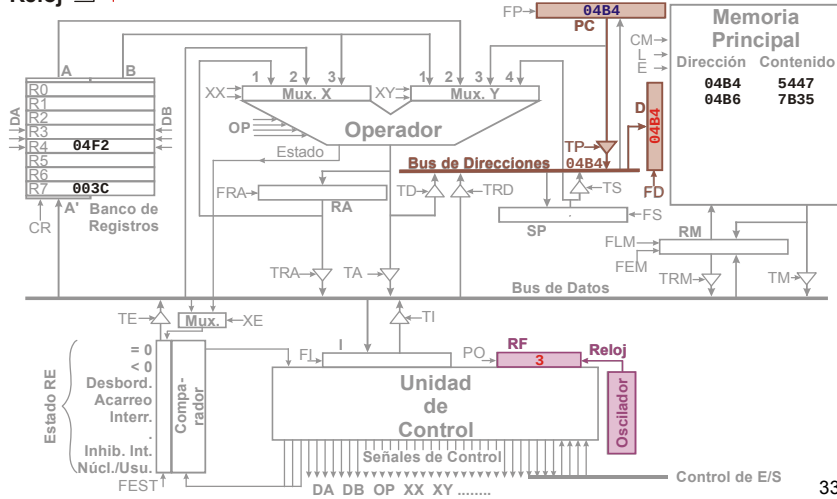
Prepara la lectura instrucción

Se realiza: $D \leftarrow PC$

Se incrementa RF: $RF \leftarrow RF + 1$

ADD .R4,.R7

Reloj



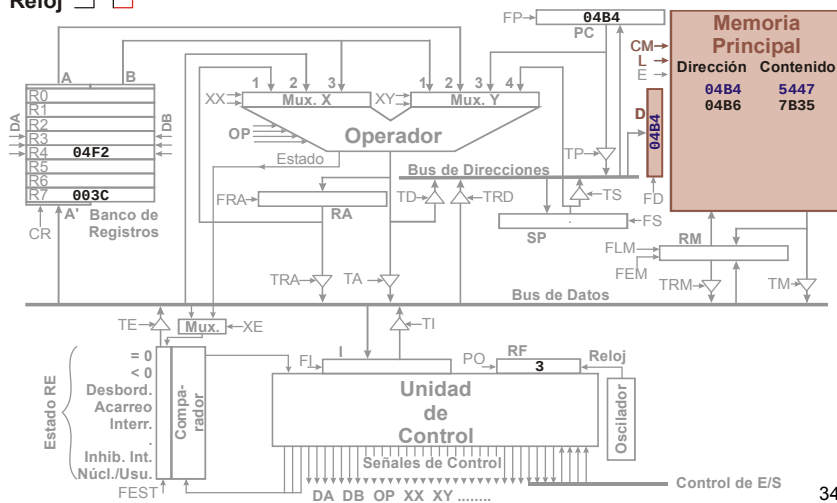
33

Lectura instrucción

Primer ciclo de memoria

ADD .R4,.R7

Reloj



34

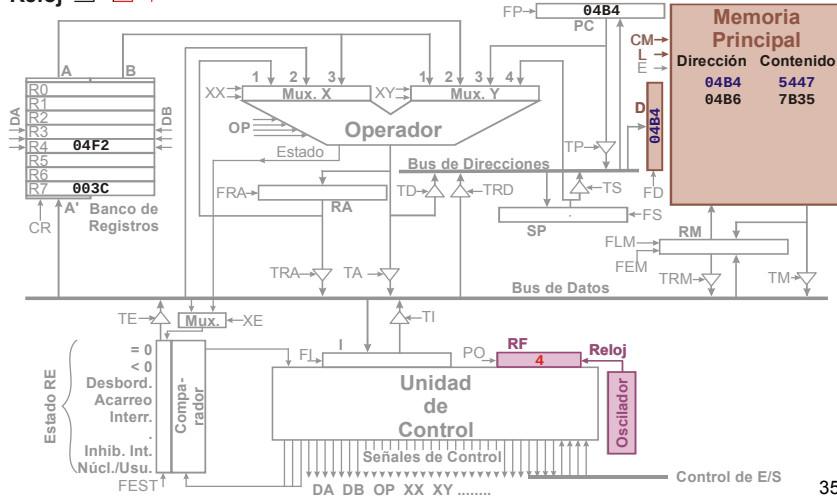
Lectura instrucción

Primer ciclo de memoria

Se incrementa RF: $RF \leftarrow RF + 1$

ADD R4,R7

Reloj



35

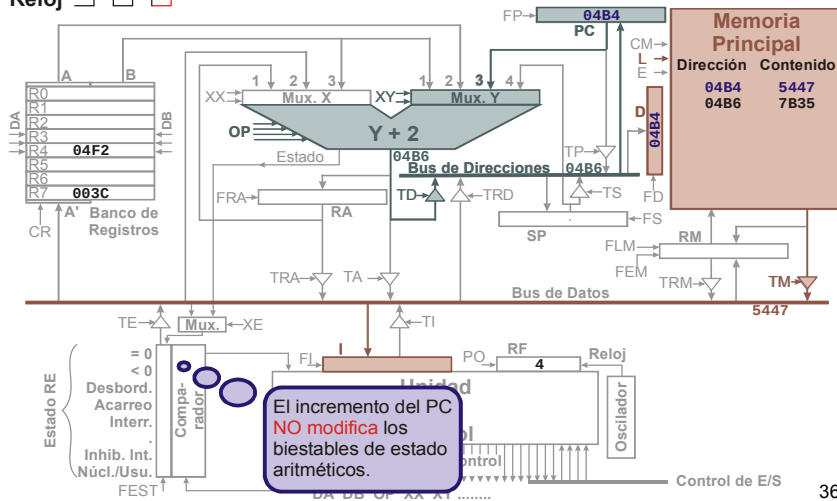
Lectura instrucción

Segundo ciclo de memoria, se prepara: $I \leftarrow M(D)$

Se prepara incremento de PC: $PC \leftarrow PC + 2$

ADD R4,R7

Reloj



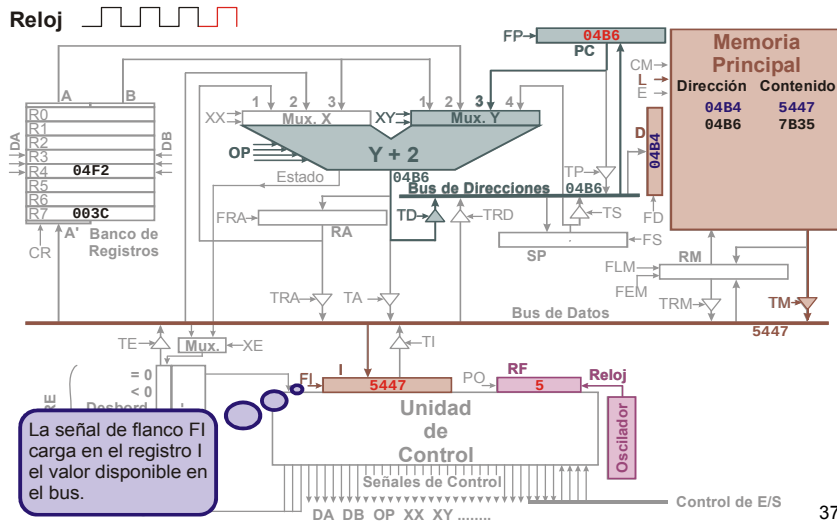
36

ADD .R4,.R7

Segundo ciclo de memoria, se realiza: $I \leftarrow M(D)$

Se realiza incremento de PC: $PC \leftarrow PC + 2$

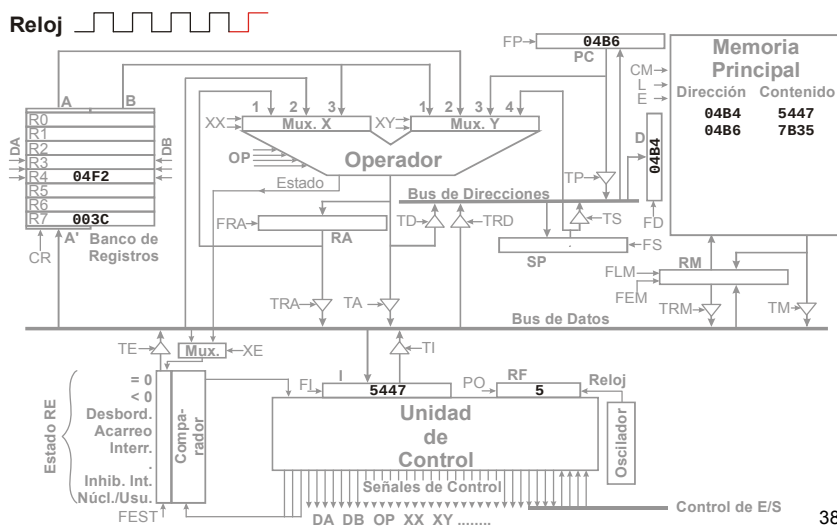
Se incrementa RF: $RF \leftarrow RF + 1$



La señal de flanco FI carga en el registro I el valor disponible en el bus.

37

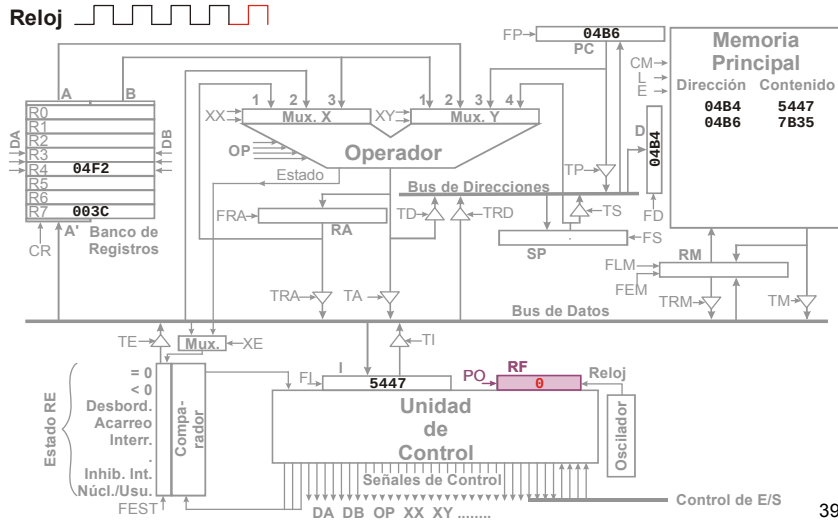
ADD .R4,.R7



38

Decodificación instrucción
 Se pone RF a 0: $RF \leftarrow 0$

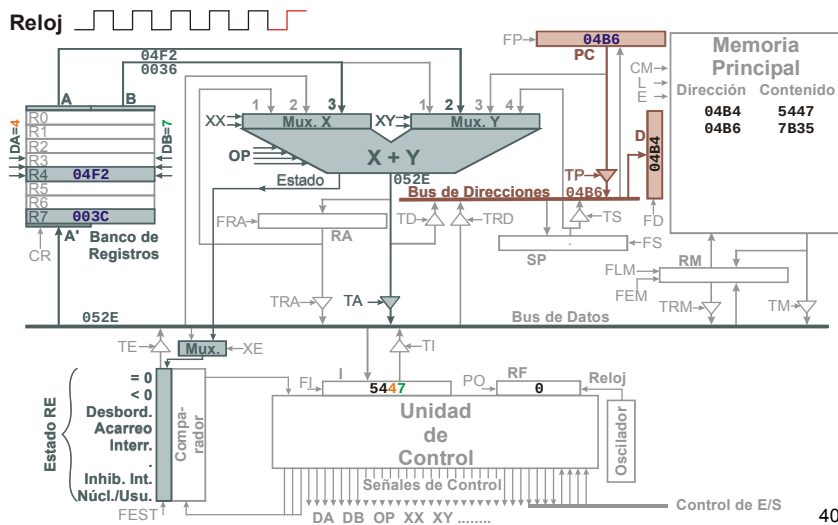
ADD .R4,.R7



39

Ejecuta instrucción y prepara lectura instrucción siguiente
 Se prepara suma: $R4 \leftarrow R4 + R7$; $RE \leftarrow$ Estado ALU
 Se prepara: $D \leftarrow PC$

ADD .R4,.R7

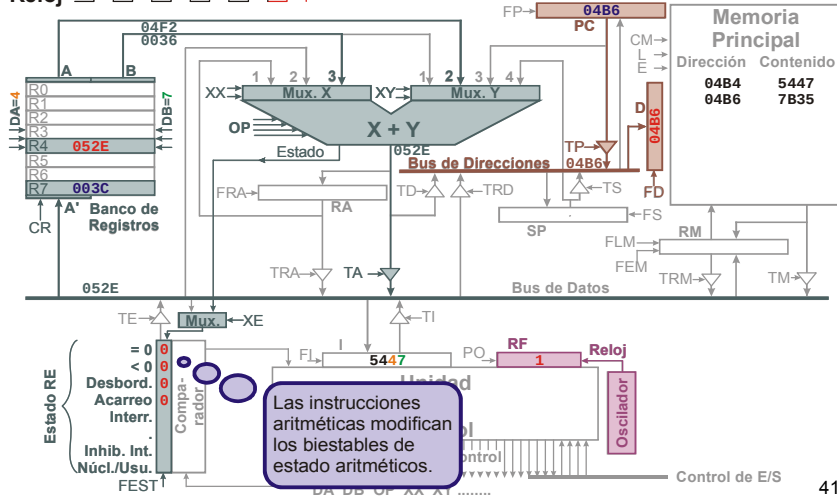


40

ADD .R4,.R7

Se realiza: $D \leftarrow PC$

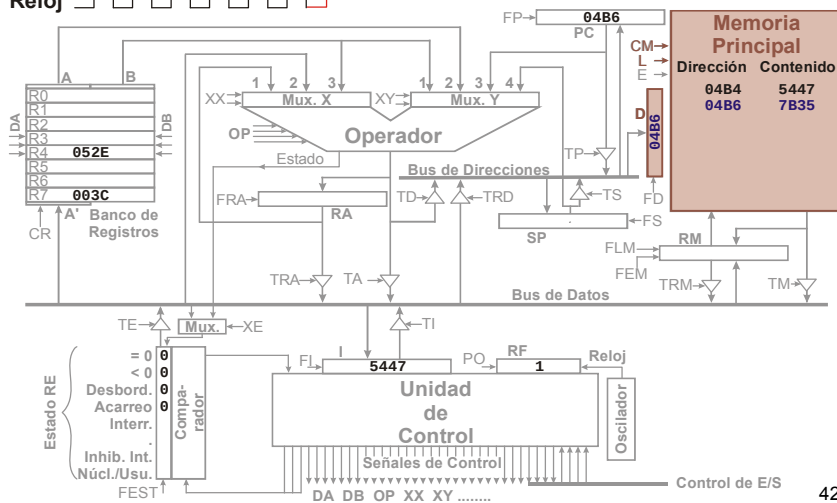
Reloj



41

ADD .R4,.R7

Reloj



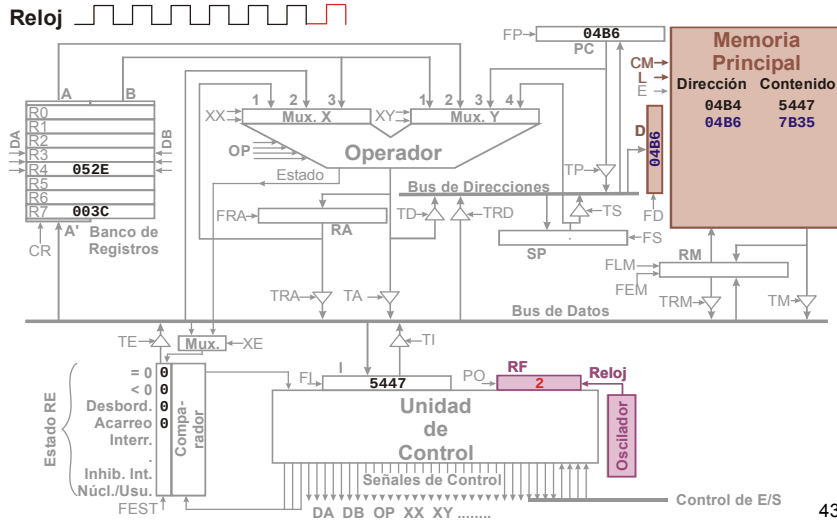
42

Lectura instrucción siguiente

Primer ciclo de memoria

Se incrementa RF: $RF \leftarrow RF + 1$

ADD R4,R7

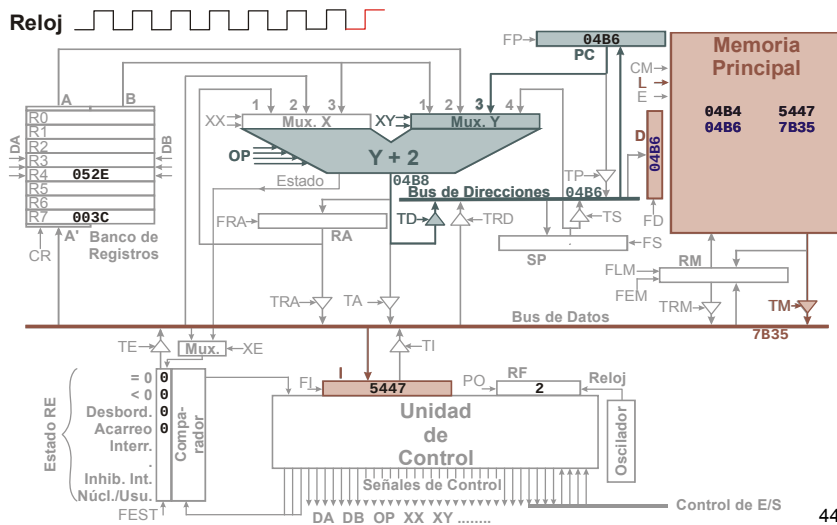


Lectura instrucción

Segundo ciclo de memoria, se prepara: $I \leftarrow M(D)$

Se prepara incremento de PC: $PC \leftarrow PC + 2$

ADD R4,R7



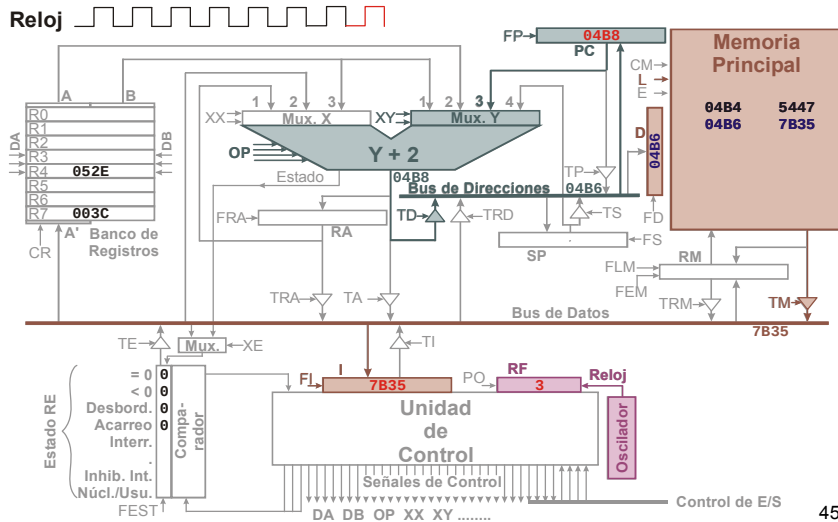
ADD .R4,.R7

Lectura instrucción

Segundo ciclo de memoria, se realiza: $I \leftarrow M(D)$

Se realiza incremento de PC: $PC \leftarrow PC + 2$

Se incrementa RF: $RF \leftarrow RF + 1$

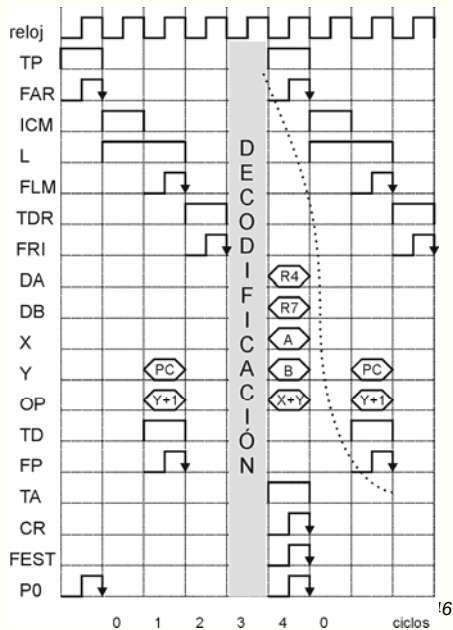


45



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

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

47



Diseño de la UC

- El diseño de la UC exige haber definido previamente
 - El repertorio de instrucciones (formatos y direccionamientos)
 - La estructura del computador
 - Conjunto de cronogramas
- La UC funciona como un TRADUCTOR
 - RI(CO, MD), Reloj, Estado, señales 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

48



Diseño de la UC

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

49



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**

50

Diseño de la UC

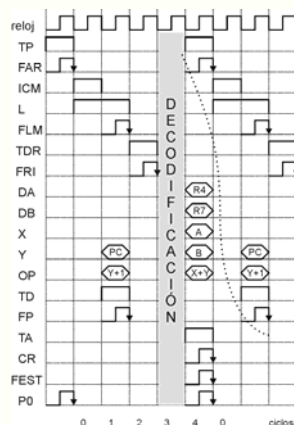
- **UC MICROPROGRAMADA**

- 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**).

51

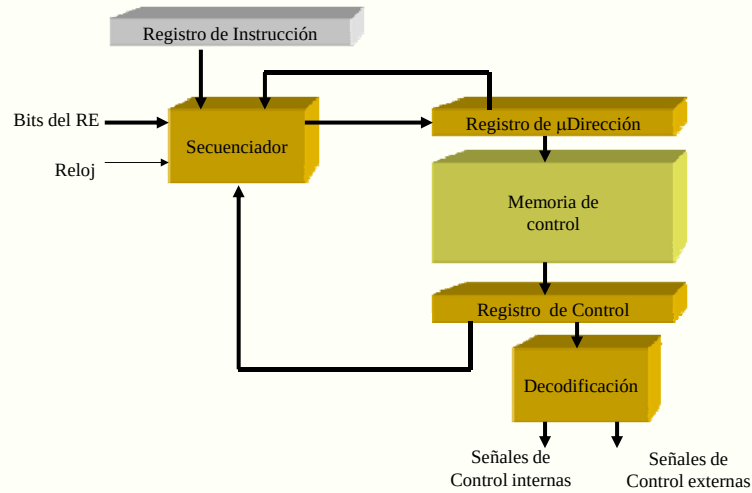
Diseño de la UC

Memoria de Control (MC)

[illegible]

52

Diseño de la UC



53

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 $\rightarrow$ 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 $\rightarrow$ cambiando sólo el microprograma!

54



Diseño de la UC

- Conceptos básicos
 - UC MICROPROGRAMADA
 - MICROINSTRUCCIÓN
 - CICLO DE MÁQUINA
 - MICROPROGRAMA

55



Conceptos básicos

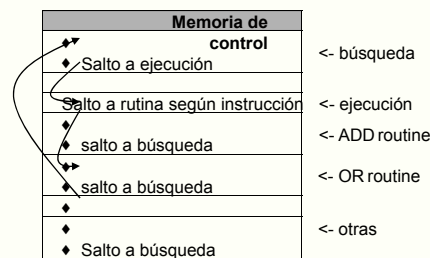
- **Microinstrucción**
 - Tira de **ceros** y **unos** que
 - representa la activación o no de cada una de las señales de control durante un ciclo de reloj
 - decide la siguiente microinstrucción que se ejecuta
 - **Cada palabra** de la memoria de control es una **microinstrucción**

56

Conceptos básicos

■ Microprograma

- Secuencia de μI 's cuya ejecución permite interpretar una instrucción
- Ejecutar una microinstrucción $\approx$ Leer una μI de la MC
- Una vez rellena la memoria de μI 's:
el control es tarea de (micro)programación
- Cambiando microprogramas se puede ejecutar otros repertorios de instrucc (simular otras archit) → ¡EMULACIÓN!



57

Conceptos básicos

■ Memoria de Control (MC)

- 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

58



Diseño de la UC

UC MICROPROGRAMADA

- Condiciones básicas:
 1. Capacidad suficiente de MC
 2. Procedimiento de correspondencia
Inst → dir MC comienzo su μ programa
 3. Secuenciamiento de μ instrucciones

59



Conceptos básicos

- **Formato de microinstrucciones**
 - **Señales de control**
 - **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 μ instrucción:** Campo que indica la siguiente μ instrucción que se ejecuta

Señales de control	Condiciones	Siguiente μ instrucción
--------------------	-------------	-----------------------------

60

Diseño de la UC

Diseño del formato de microinstrucción

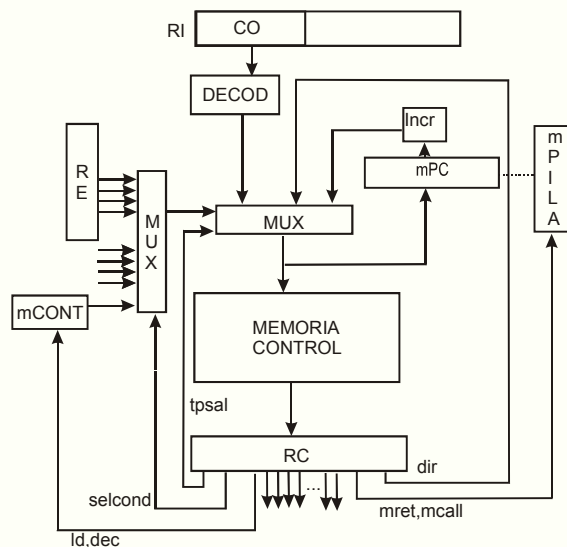
- Comenzar con la lista completa de señales de control
 - Agrupar las señales de control con función similar en un mismo **CAMPO**
 - Controlar que una misma señal no esté en diferentes campos
 - Campo de **CONTROL DE SECUENCIAMIENTO**: indica la siguiente μI que se ha de ejecutar
 - Secuenciamiento: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

61

Diseño de la UC

Estructura completa de la UC mprogramada

- microbucles
- microsubsutinas



62



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

63



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 comunicarse con los periféricos?
 - ...

64



Control de excepciones

- Excepción [Overflow, división por cero, etc.]
 - Evento no planificado que detiene la ejecución de un programa: **interno**
 - Salvar el PC y el Reg. Estado actuales en pila
 - 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

65



Control de excepciones

Tipos de excepciones

- **Interrupciones:**
 - Causadas por eventos **externos** (peticiones de periféricos).
 - Asíncronas a la ejecución del programa (en cualquier instante)
 - Se atienden entre instrucciones (fetch)
 - Obligan a suspender la ejecución del programa de usuario

66



Control de excepciones

Tipos de excepciones

■ Traps:

- 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

67



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 reg. de estado del programa de usuario
 - Retorna el control a usuario (salvo que se aborte)

68



Control de excepciones

◆ Ej.: Interrupciones Vectorizadas:

1. Cada excepción puede tener una dirección diferente asociada
 2. Detectada la excepción → la CPU calcula para esa **excepción** la dirección que se lleva al PC (depende del periférico, que pone un VECTOR en bus).
- ¿Cuándo atenderlas?
 - Entre Instrucciones: antes de comenzar el FETCH

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