

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

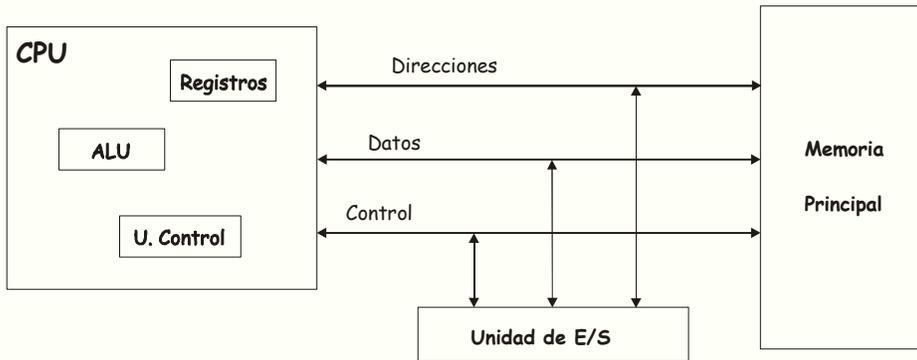
Bibliografía

- 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

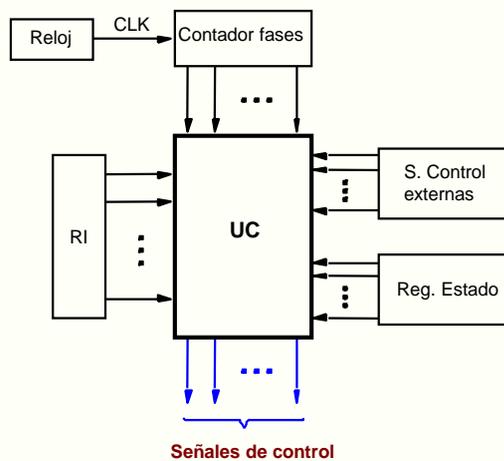
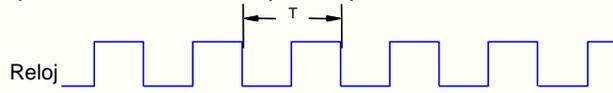


Figura Entradas y salidas de la UC.

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 \rightarrow AR
2. $M(AR) \rightarrow$ Reg; lectura

lectura: MEMRQ y RD
Reg o (DR)

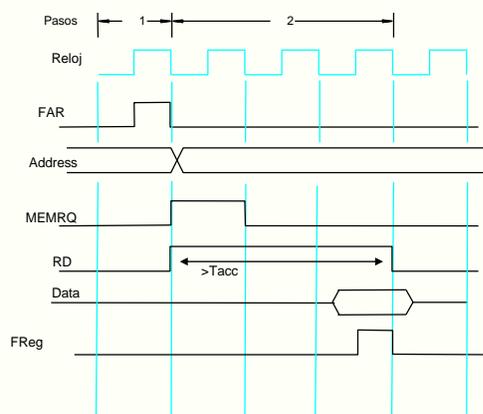


Figura . Temporización de la lectura en memoria

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

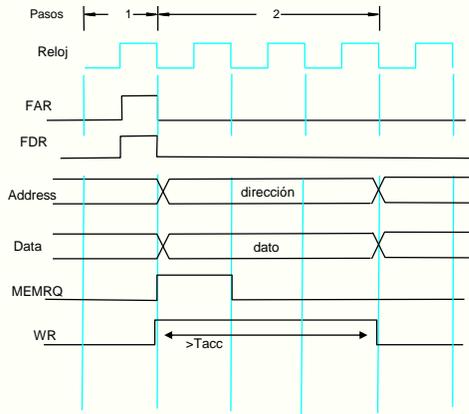


Figura . Temporización de la escritura en memoria

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

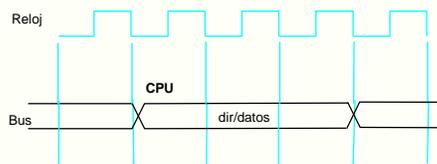


Figura . Ciclo de bus

Introducción

Evaluación del rendimiento

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

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

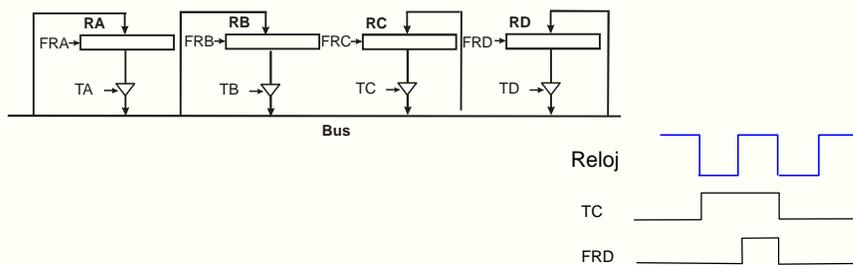
- **OPERACIONES ELEMENTALES** (microoperaciones): operaciones (**realizables directamente por el hardware**) en que la UC divide cada una de las fases de ejecución de una instrucción
- Las operaciones elementales se realizan por la UC mediante la **activación de señales de control**
- Cada microoperación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



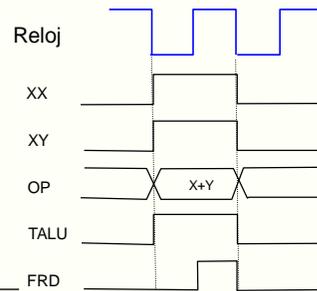
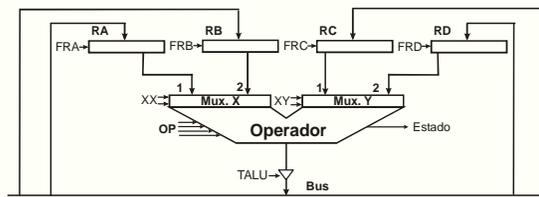
Operaciones Elementales

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

$$A+C \rightarrow D$$



Unidad de Control

20

Operaciones Elementales

Reglas de agrupación de μ operaciones

- Respetar orden preestablecido 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)

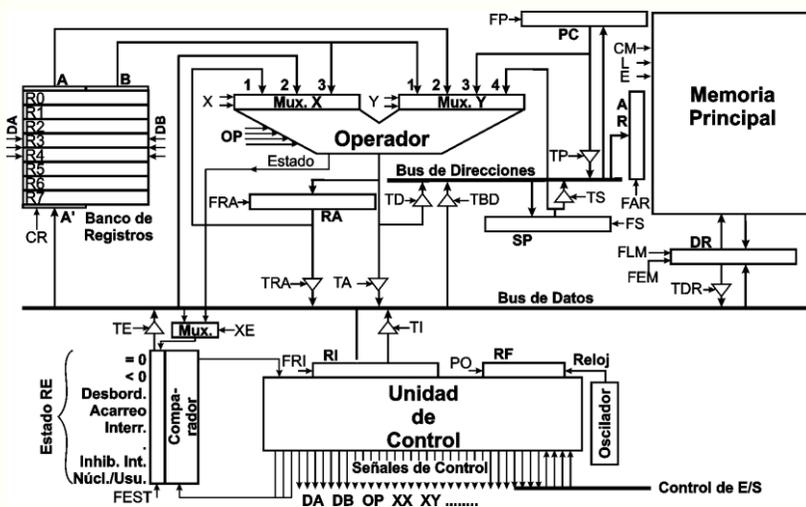
Unidad de Control

21

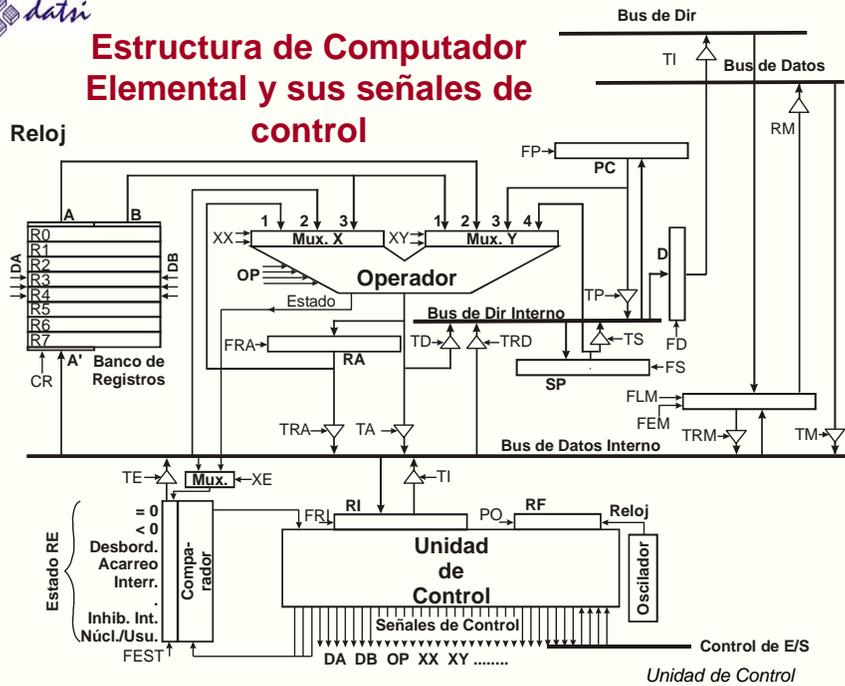
Unidad de Control

- 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

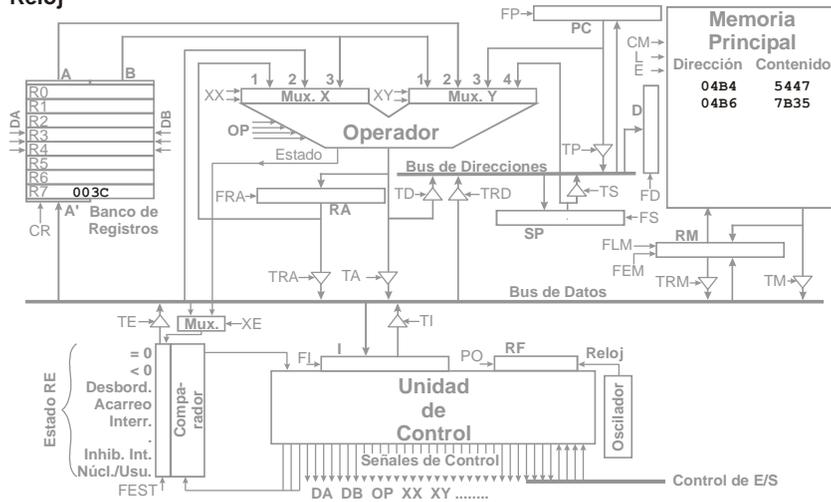


Unidad de Control

- Índice
 - 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

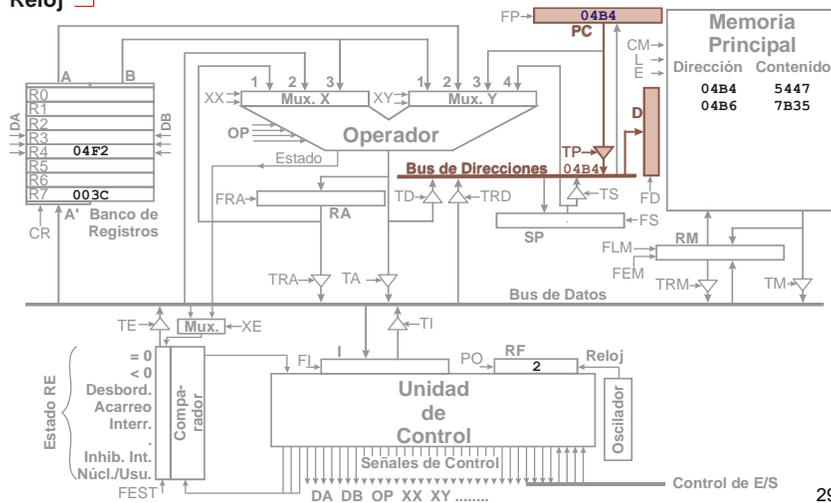


Prepara la lectura instrucción

Se prepara: $D \leftarrow PC$

ADD .R4,.R7

Reloj

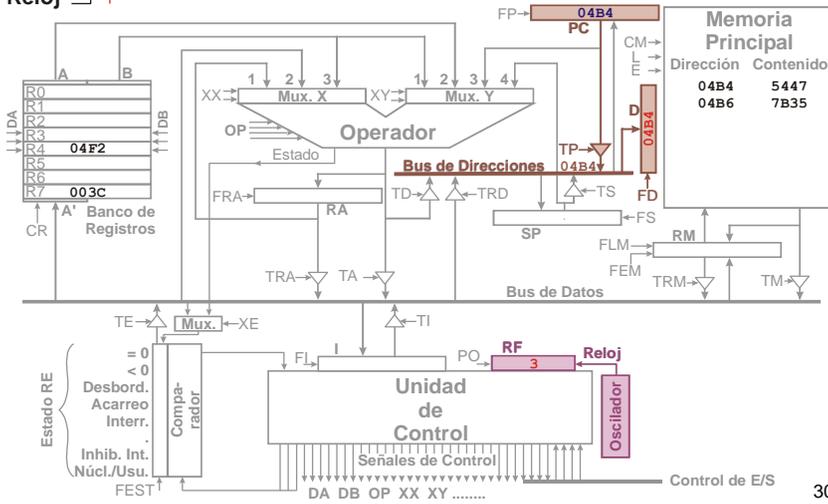


Prepara la lectura instrucción

ADD .R4..R7

Se realiza: $D \leftarrow PC$
 Se incrementa RF: $RF \leftarrow RF + 1$

Reloj



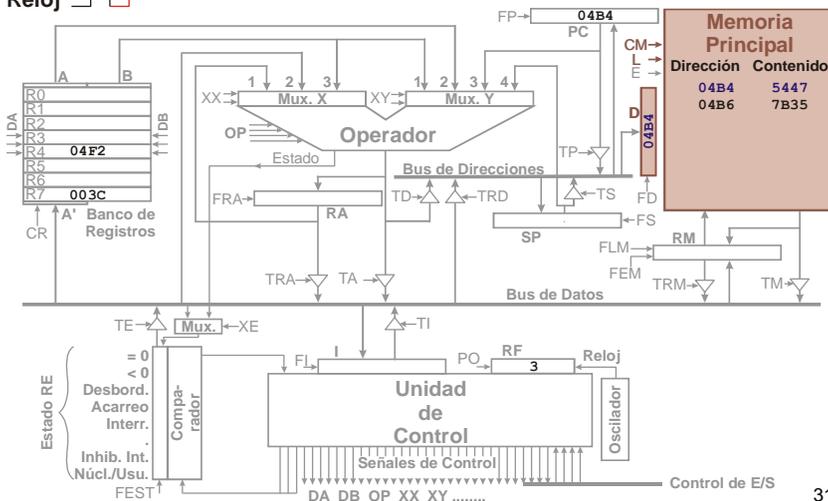
30

Lectura instrucción

ADD .R4..R7

Primer ciclo de memoria

Reloj



31

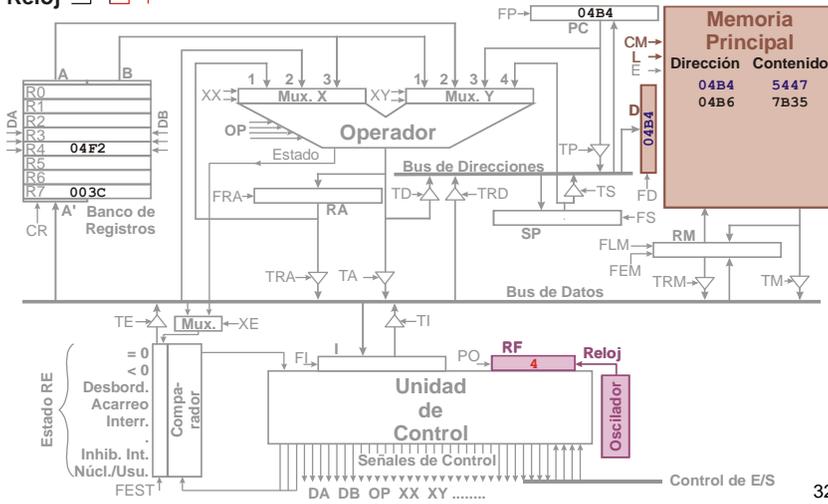
Lectura instrucción

ADD .R4..R7

Primer ciclo de memoria

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

Reloj



32

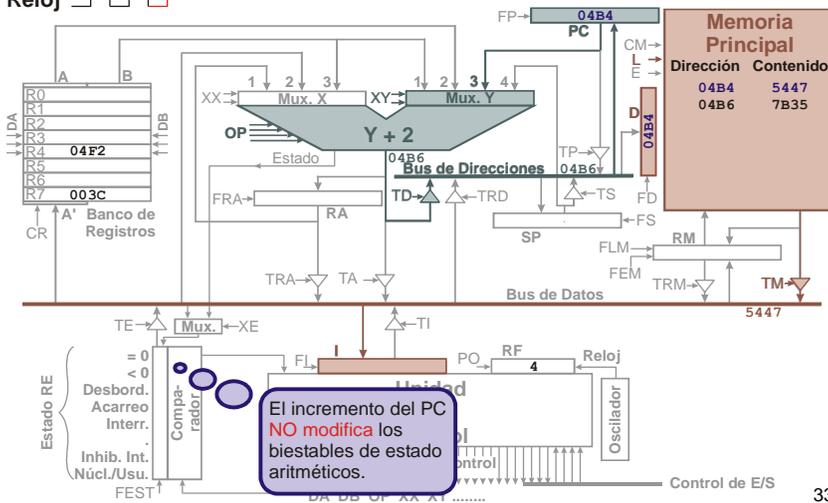
Lectura instrucción

ADD .R4..R7

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

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

Reloj



33

ADD .R4..R7

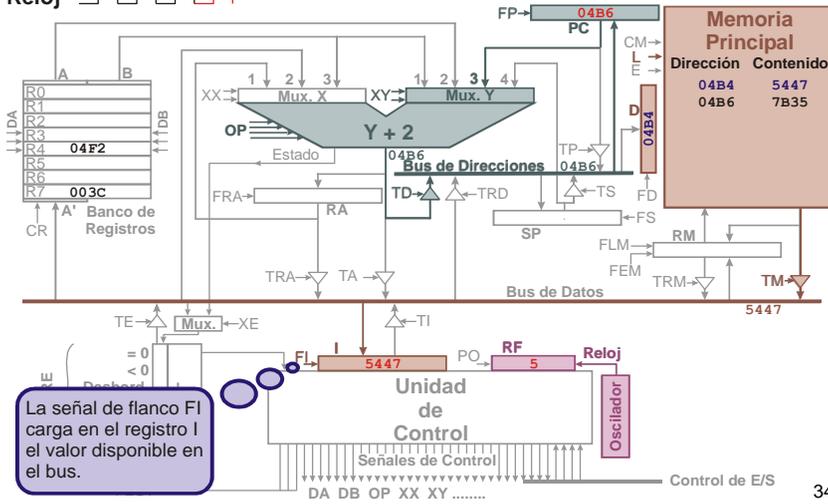
Lectura instrucción

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

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

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

Reloj

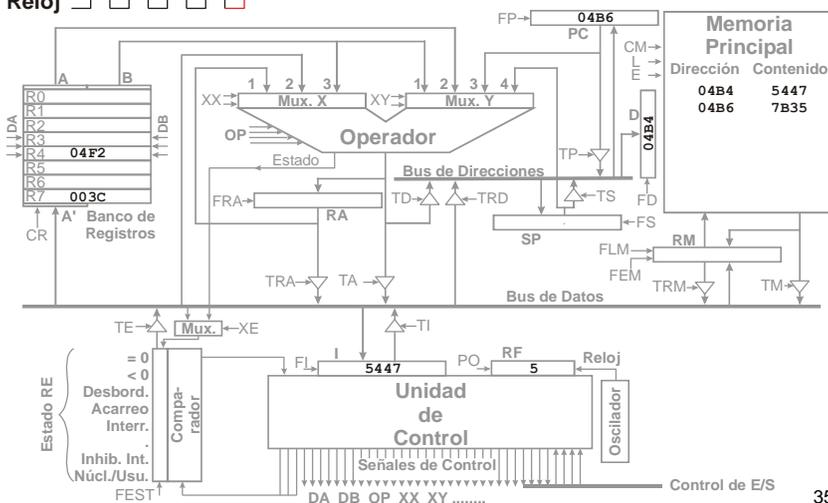


34

ADD .R4..R7

Decodificación instrucción

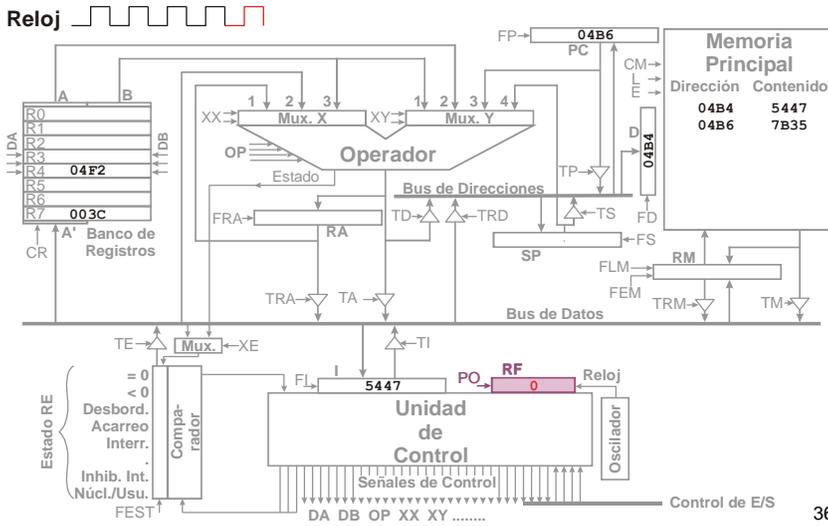
Reloj



35

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

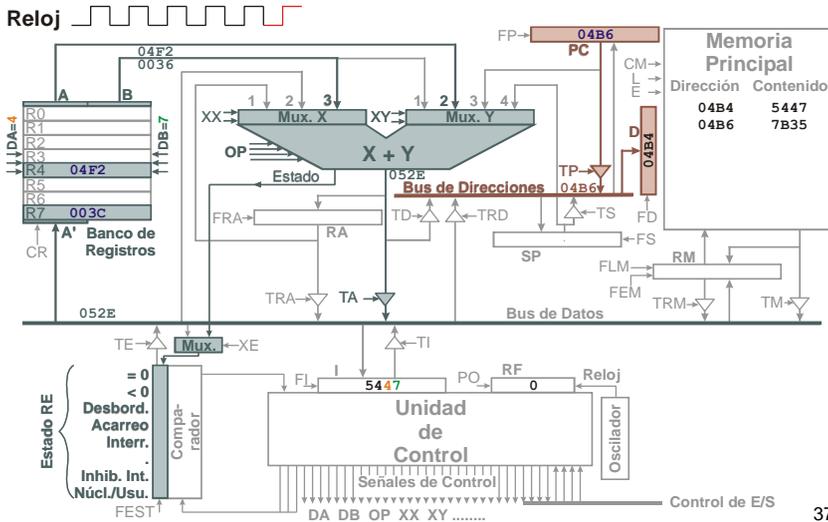
ADD .R4,.R7



36

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



37

Ejecuta instrucción y prepara lectura instrucción siguiente

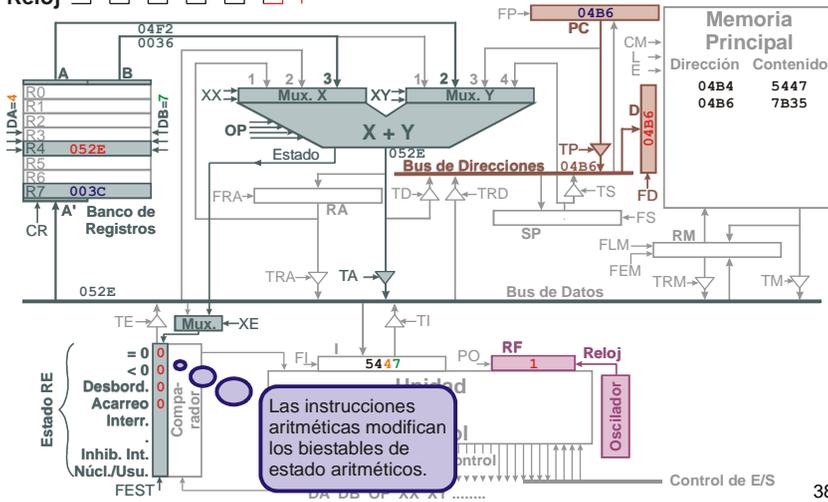
ADD .R4,.R7

Se realiza suma: $R4 \leftarrow R4 + R7$; RE \leftarrow Estado ALU

Se realiza: $D \leftarrow PC$

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

Reloj 



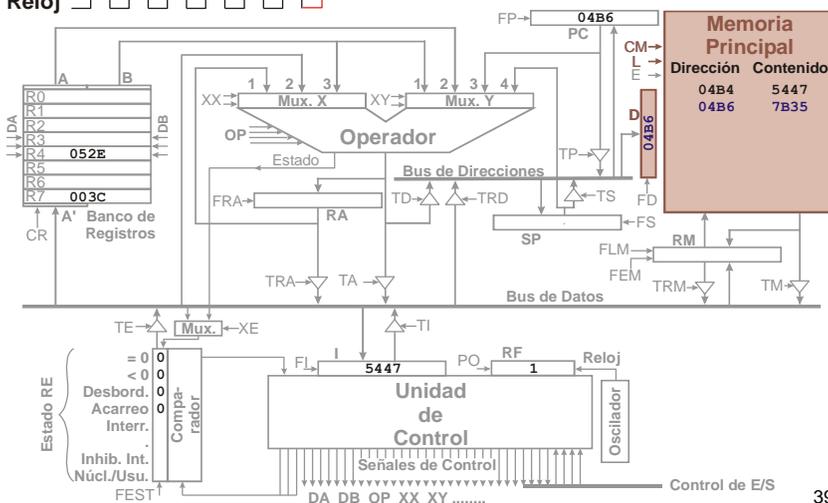
38

Lectura instrucción siguiente

ADD .R4,.R7

Primer ciclo de memoria

Reloj 



39

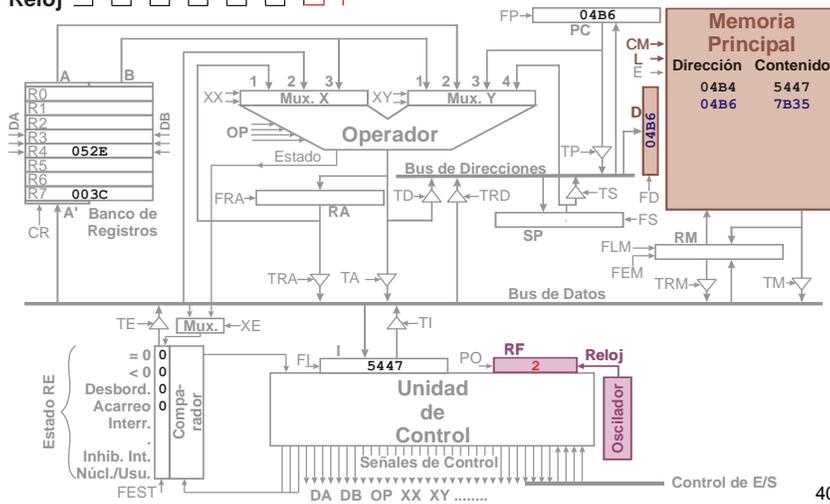
ADD .R4..R7

Lectura instrucción siguiente

Primer ciclo de memoria

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

Reloj 



40

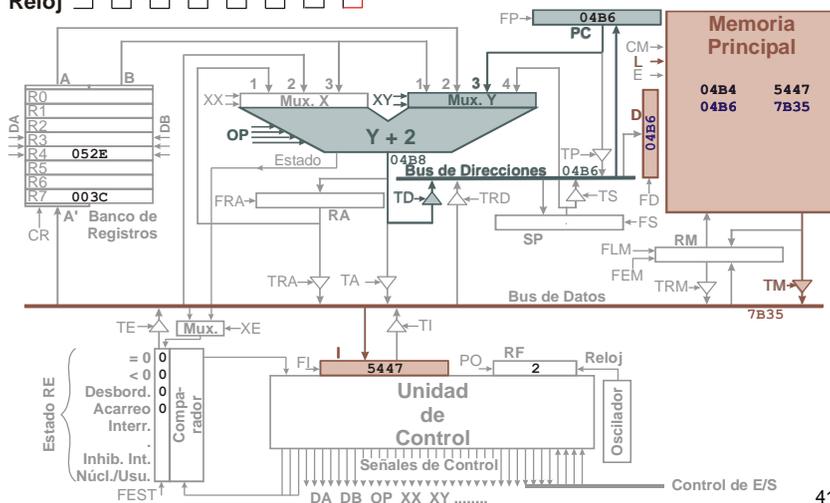
ADD .R4..R7

Lectura instrucción

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

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

Reloj 



41

ADD .R4,.R7

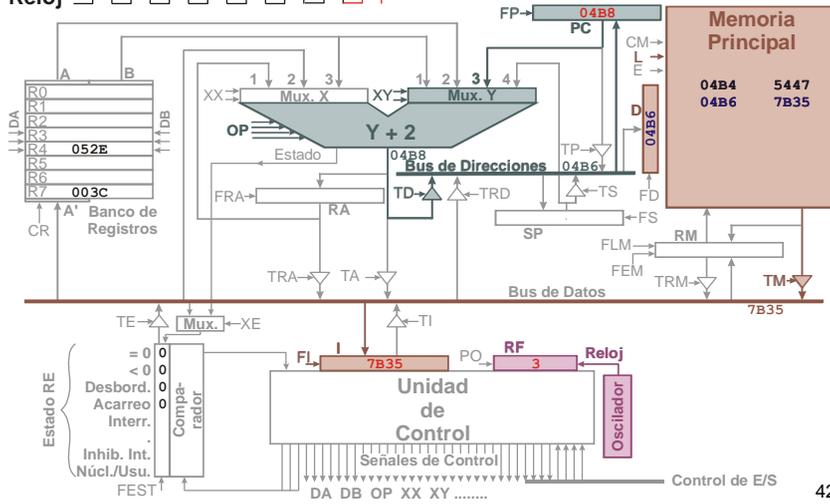
Lectura instrucción

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

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

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

Reloj 

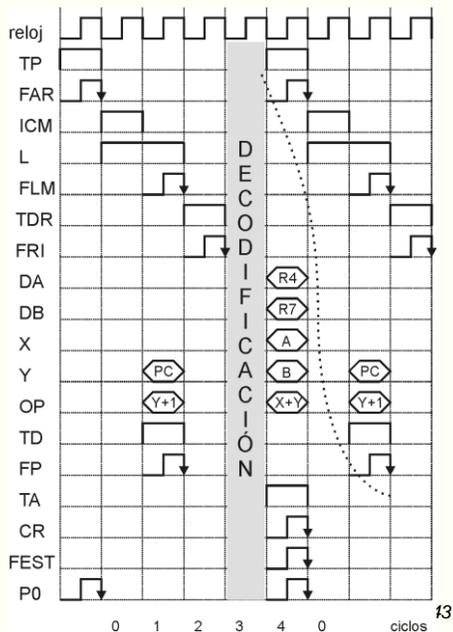


42



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)



13

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

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

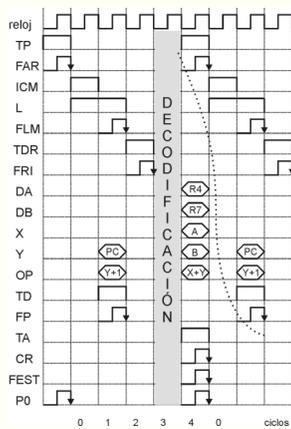
Diseño de la UC

- 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

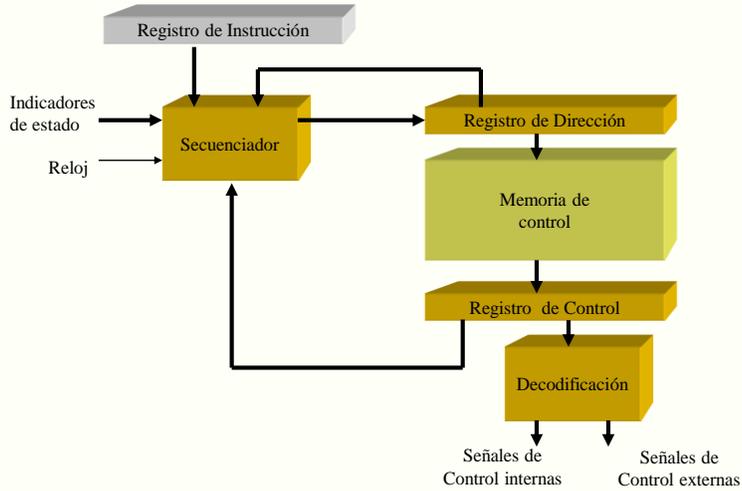


Memoria de Control

	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	1	1	0	1	0	1	0	0	0	0	0
1	0	0	0	1	1	0	0	0	1	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	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	1	0	0
0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	1	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
1	0	1	0	1	1	0	0	1	0	0	1	0	1	0	0	0	0	1	0	0
x	x	x	x	x	x	0	0	1	1	0	1	0	0	1	1	0	0	0	1	1
x	x	x	x	x	x	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
x	x	x	x	x	x	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1
x	x	x	x	x	x	0	1	1	0	0	1	1	0	0	0	0	0	0	0	1
x	x	x	x	x	x	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0
x	x	x	x	x	x	0	1	1	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	1	0	0
x	x	x	x	x	x	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0



Diseño de la UC



Diseño de la UC

UC CABLEADA	UC MICROPROGRAMADA
<p>Ventajas</p> <ul style="list-style-type: none"> ● Alta velocidad de funcionamiento ● Implementaciones más pequeñas (reducido número de componentes) 	<p>Ventajas</p> <ul style="list-style-type: none"> ● Son sistemáticas con un formato bien definido ● Pueden ser fácilmente modificables durante el proceso de diseño
<p>Inconvenientes</p> <ul style="list-style-type: none"> ● Difícil realizar modificaciones en el diseño: modificación → rediseño ● No tienen estructura común 	<p>Inconvenientes</p> <ul style="list-style-type: none"> ● Requieren más componentes para su implementación ● Tienen 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
<p>■ Técnica de diseño favorita en las arquitecturas RISC</p>	<p>■ Emulación</p> <ul style="list-style-type: none"> ● 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

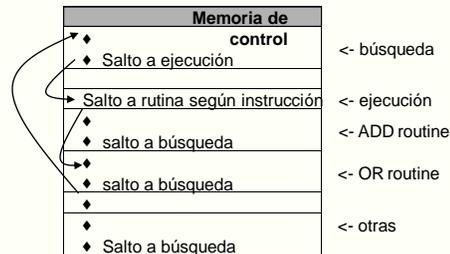
Conceptos básicos

- **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 \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!**



Unidad de Control

54

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

Unidad de Control

55

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

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 μ instrucción:** Campo que indica la siguiente μ instrucción a ejecutar

Señales de control	Tipo Sec.	Cond	Siguiente μ instr
--------------------	-----------	------	-----------------------

Diseño de la UC

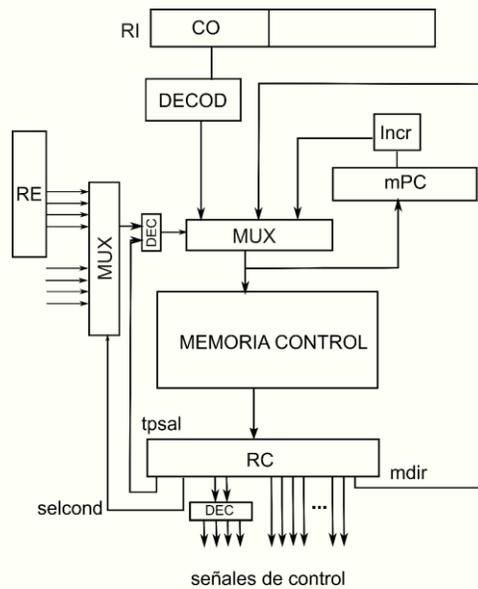
Diseño del formato de microinstrucción

- 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 ml 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 ml
 - 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

- micros saltos condicionales



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

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)