Arquitectura de Computadoras

CURSO 2022

Turno:

Clase 4

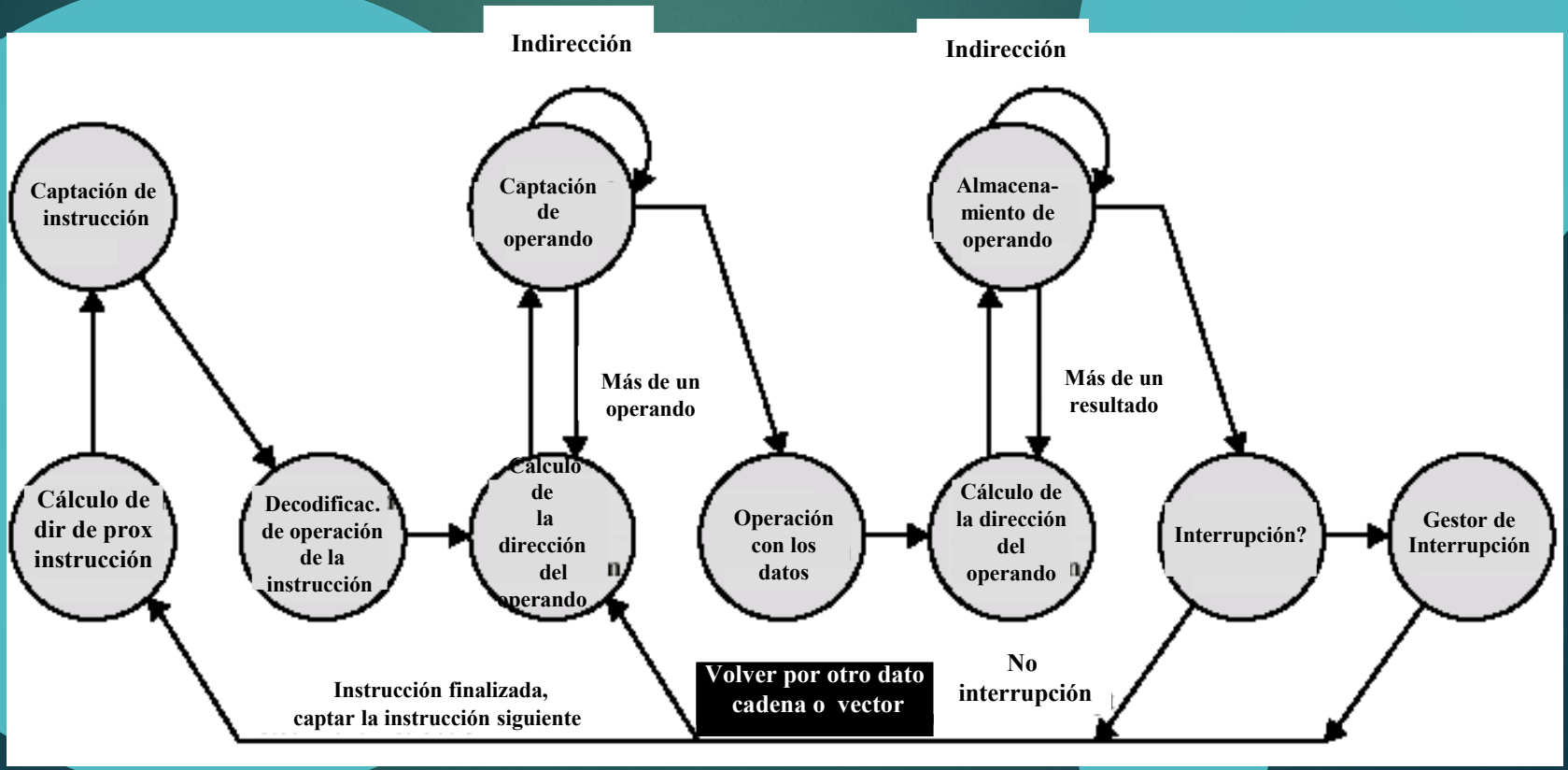
Resumen clase 4

Segmentación del cauce

- Análisis del nanoMips
  - Introducción
  - Formato de instrucción
  - Modos de direccionamientos
  - Repertorio de instrucciones
  - Rutas de datos, instrucciones y control
  - Diagrama de estados
- Segmentación de instrucciones
- Segmentación en el nanoMIPS
- Conflictos y soluciones en la segmentación

Diagrama de estados del ciclo de instrucción

En la figura siguiente se muestra el diagrama de estados de una CPU convencional. Este diagrama de estados va a servir de referencia para el procesador que se va a describir a continuación (nanoMIPS).



NanoMIPS

Introducción al nanoMIPS

- Para algunos de los temas a tratar a continuación, es mejor usar un modelo de procesador más sencillo del que se usó hasta ahora (8086 / MSX88).
- Se va a usar un modelo de procesador simplificado basado en el procesador MIPS, que es un procesador comercial tipo RISC.

NanoMIPS

Introducción al nanoMIPS

El nanoMIPS tiene estas características principales:

- Palabra de memoria de 32 bits
- Espacio de direcciones de 64 bits (virtual)
- 32 (31 efectivos) Registros de propósito general de 64 bits (R0..R31).
- El registro R0 es 0 (es decir, si referenciamos al registro R0 nos devuelve 0).
- Como son 32 registros, se requieren 5 bits para identificarlo.

NanoMIPS

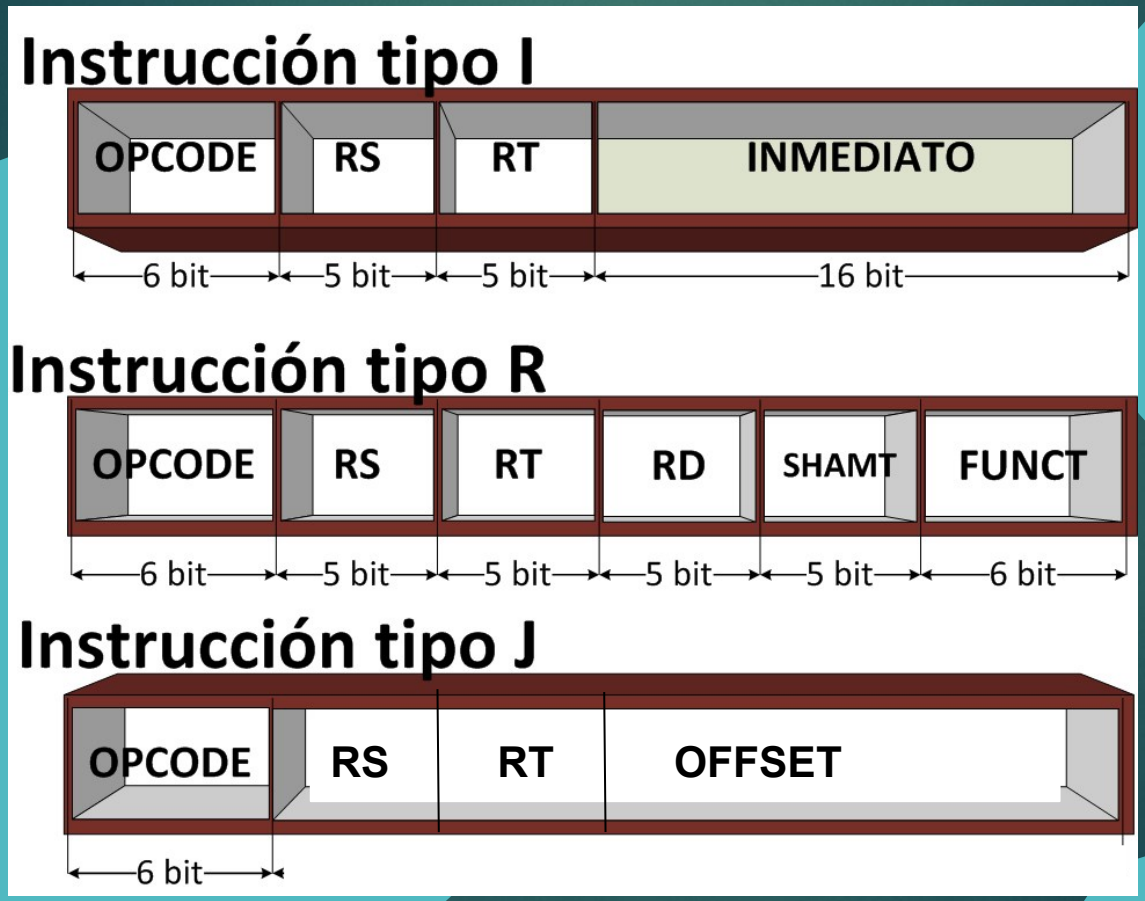
Formato de instrucción

El nanoMIPS dispone de 3 tipos de Instrucciones:

- Tipo I: instrucciones de acceso a memoria, únicamente las de carga y almacenamiento (“LOAD” y “STORE”).
- Tipo R: instrucciones aritméticas y lógicas, únicamente registro a registro (no admite operaciones con memoria).
- Tipo J: instrucciones de salto
- El acceso a la memoria se hace únicamente con las instrucciones tipo I. Por esta razón a este tipo de máquinas se las identifica como máquina tipo “LOAD/STORE”.
- El formato es muy regular, porque:
  - Las instrucciones tienen todas el mismo tamaño (32 bits).
  - Los campos con referencias son siempre los mismos.

NanoMIPS

Formato de instrucción



Load/store

Aritm./lógicas

Salto cond./incond.

NanoMIPS

Modos de direccionamiento

El nanoMIPS dispone formalmente de 2 modos de direccionamiento, pero si se consideran 2 casos especiales, se obtienen 2 modos más (4 en total).

- Inmediato: dato de 16 bits.
- Indirecto con desplazamiento: suma de registro más un desplazamiento (distinto de 0).
- Indirecto vía registro: si el desplazamiento es 0 el direccionamiento es indirecto vía registro.
- Directo absoluto: Si el desplazamiento es vía registro R0 (que vale 0), entonces se obtiene directo absoluto.

NanoMIPS

Repertorio de instrucciones

Para analizar el comportamiento de este procesador, se va a considerar un repertorio básico de 8 instrucciones: 2 de movimiento de datos con memoria, 5 aritméticas y logicas, y 1 de salto condicional.

	Instrucción	Pseudocódigo	Descripción
I	**LW**	LW RT, desp(RS)	Carga registro RT desde memoria
	SW	SW RT, desp(RS)	Almacena en memoria desde registro RT
R	**ADD**	ADD RD, RS, RT	Suma palabras en registros RS y RT, resultado en RD
	SUB	SUB RD, RS, RT	Resta palabras en registros RS y RT, resultado en RD
	AND	AND RD, RS, RT	AND de palabras en registros RS y RT, resultado en RD
	OR	OR RD, RS, RT	OR de palabras en registros RS y RT, resultado en RD
	SLT	SLT RD, RS, RT	Pone 1 en RD si RS es menor o igual que RT
J	**BEQ**	BEQ RS, RT, destino	Salta a 'destino' si RS es igual a RT

NanoMIPS

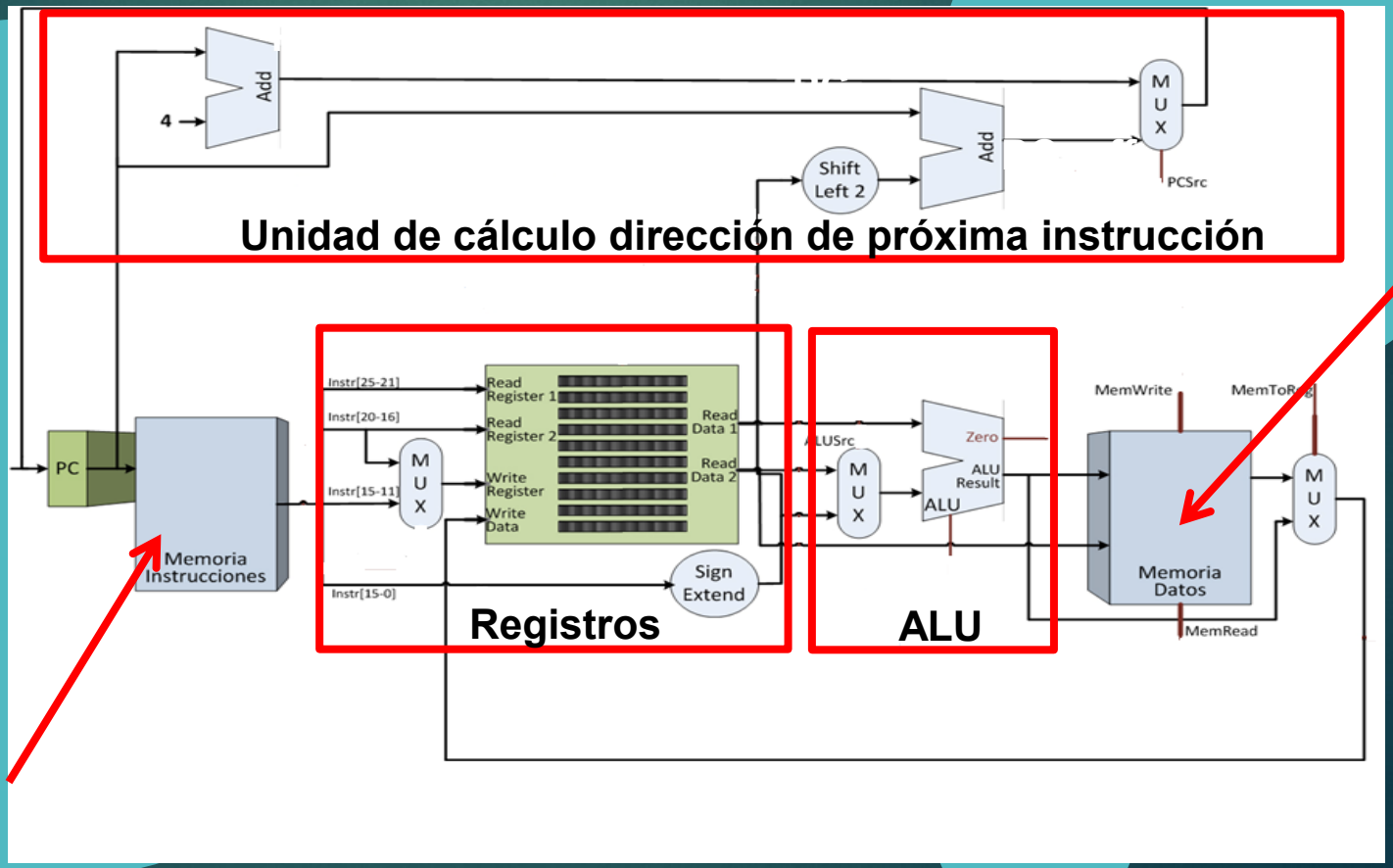
Repertorio de instrucciones

- Instrucciones tipo I: Load/store
  - RS: registro base
  - RT: Registro de fuente o destino de la transferencia
  - Desp: offset
- Instrucciones tipo R: Aritmético/lógicas
  - RS: operando 1
  - RT: operando 2
  - RD: resultado
- Instrucciones tipo J: de salto
  - RS: operando 1
  - RT: operando 2
  - Destino: es un desplazamiento respecto del PC (16 bits)

NanoMIPS

Ruta de datos e instrucciones

En la imagen siguiente se muestra la ruta de los datos e instrucciones dentro y fuera de la CPU



NanoMIPS

Ruta de datos e instrucciones

La imagen anterior muestra la ruta de los datos e instrucciones del nanoMips. Se pueden identificar:

- Banco de registros:
  - Está compuesto por 32 registros (el R0=0)
  - Se pueden leer 2 registros al mismo tiempo (doble entrada)
  - Se puede escribir en 1 solo registro
- ALU:
  - Es una unidad de cálculo típica de 2 entradas y 1 salida
  - En la entrada se pueden seleccionar diferentes fuentes de datos
- Unidad de cálculo de dirección de próxima instrucción:
  - Calcula el próximo valor con el que se carga el PC
  - Puede ser el de la instrucción consecutiva (PC+4), o sumando un desplazamiento (instrucción de salto con desplazamiento).

NanoMIPS

Ruta de datos e instrucciones

También se puede apreciar que existen 2 bancos de memoria:

- Memoria de instrucciones:
  - Solo contiene el programa a ejecutar
  - Se accede únicamente para lectura, en la fase de búsqueda de la instrucción, y accedida solo a través del PC.
- Memoria de datos:
  - Contiene los datos a leer o escribir
  - Solo se accede a través de las instrucciones LOAD y STORE

NanoMIPS

Ruta de datos, instrucciones y control

La Unidad de control (CU) es una unidad relativamente sencilla que captura la instrucción y a partir de ella genera las señales de control requeridas por los demás bloques operativos.

En particular, la CU necesita el campo del código de operación (OPCODE) para determinar que tareas deberá realizar para completar la instrucción corriente.

El resto de los campos se usan para las identificaciones restantes.

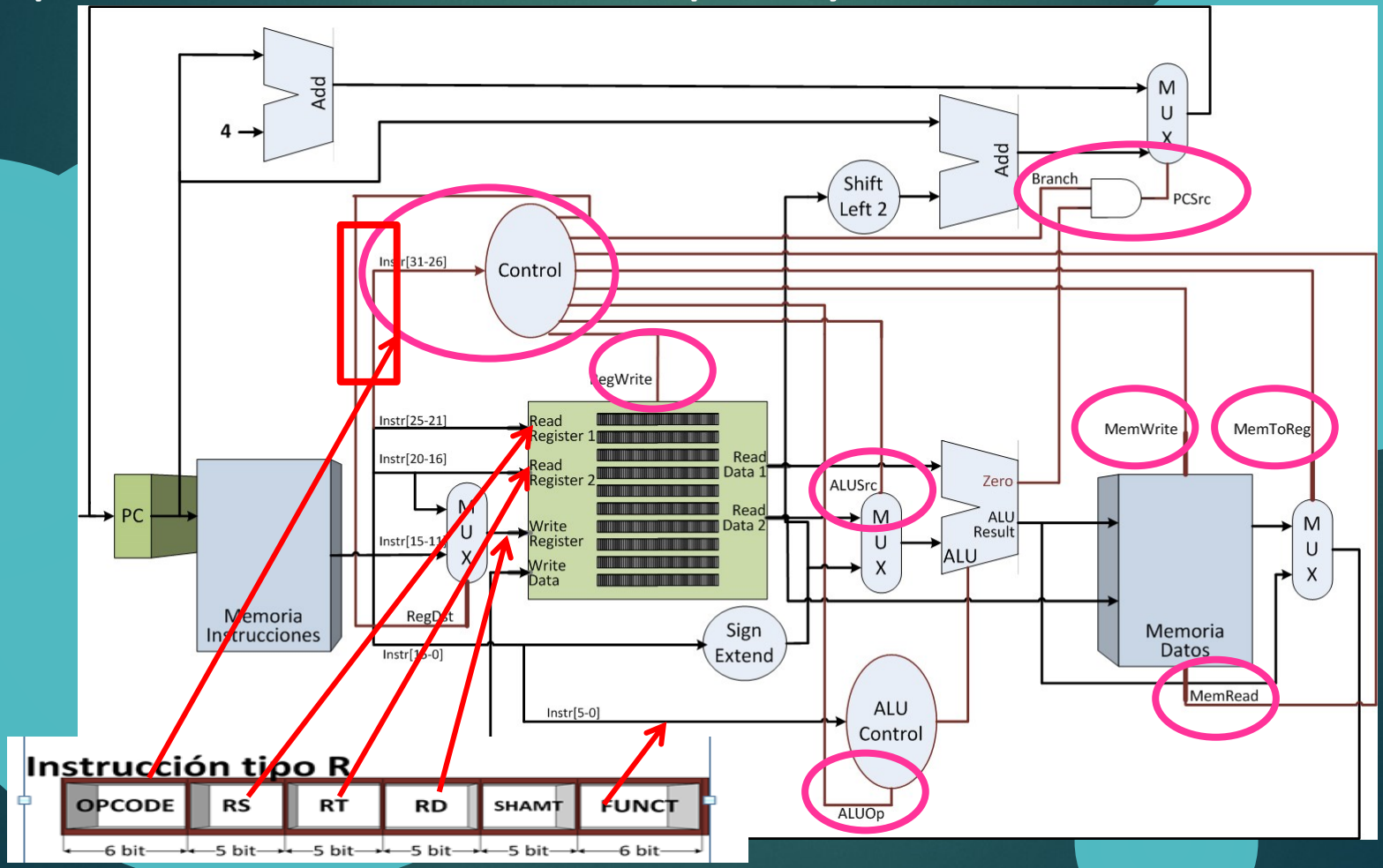
Por ejemplo:

- El campo RS selecciona 1 registro en el banco de registros
- Idem los campos RT y RD (cuando corresponda)
- El resto de los campos se usan para completar las órdenes a dar a las Unidades funcionales restantes (por ejemplo la ALU).

NanoMIPS

Ruta de datos, instrucciones y control

En la imagen siguiente se muestra la asociación entre los campos de una instrucción tipo R y su uso dentro de la CPU.



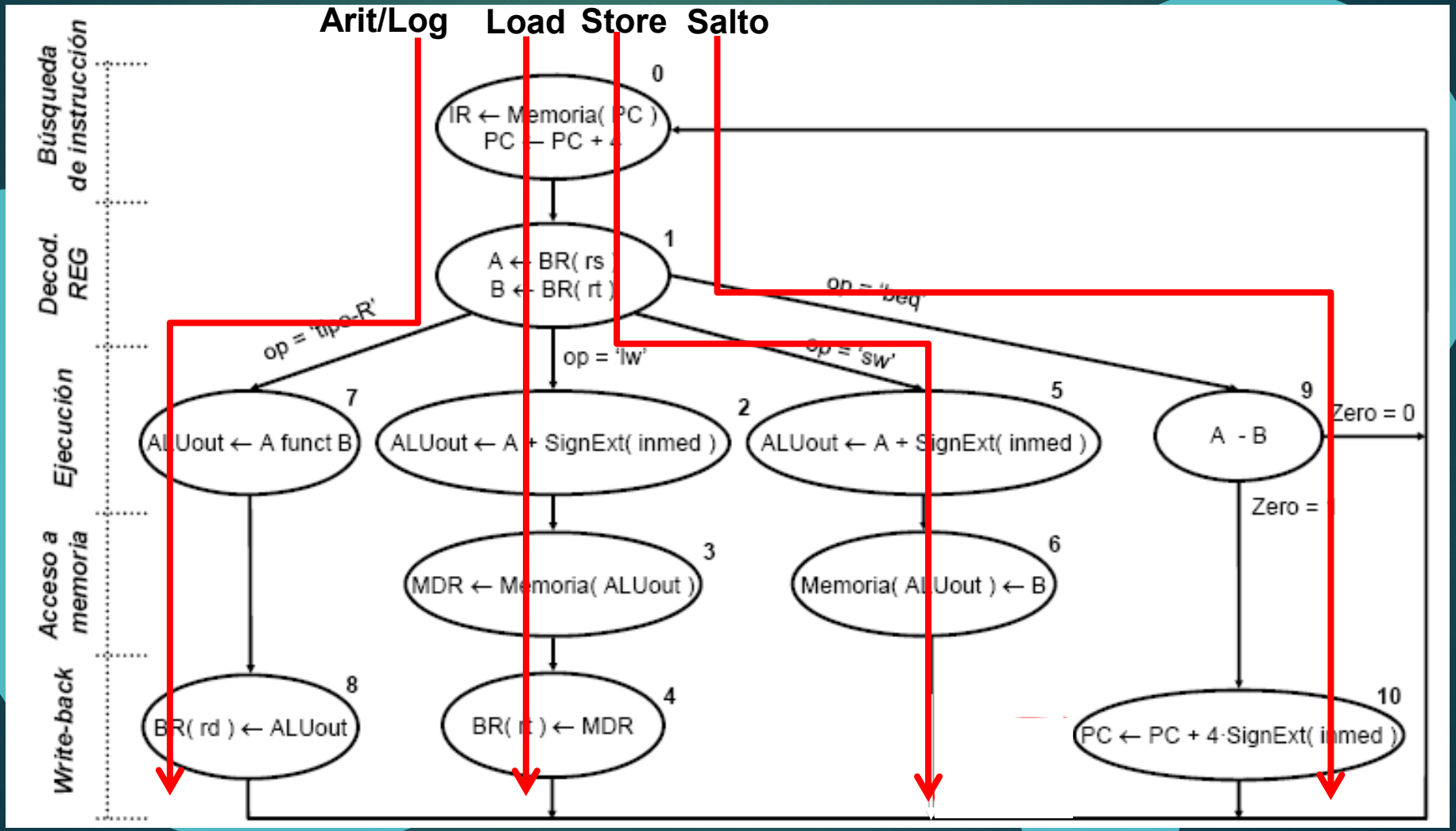
NanoMIPS

Ruta de datos, instrucciones y control

En la imagen anterior se puede apreciar que la Unidad de control (“Control”), a partir de los bits del campo OPCODE (bits 31-26) genera las señales de control para las demás unidades, por ejemplo RegWrite, ALUSrc, MemWrite, MemToReg, etc.

NanoMIPS

Diagrama de estados



NanoMIPS

Diagrama de estados

Tiene una secuencia de instrucción compuesta por 5 estados.

Fase 1 - Búsqueda de instrucción (F, Fetch):

- ocurre en todas las instrucciones
- Busca y lee la instrucción en la memoria (de instrucciones)
- Actualizar el PC (sumar 4 al valor actual del PC): cada instrucción ocupa 4 bytes, por lo que la próxima instrucción consecutiva está en la dirección actual + 4.

Fase 2 - Decodificación (D, Decode) y acceso a registros:

- ocurre en todas las instrucciones
- Se decodifica la instrucción (del campo CODOP)
- Se accede al banco de registros
- Opcional: Extensión del signo del offset para cálculo de la dirección efectiva

NanoMIPS

Diagrama de estados

Fase 3 - Ejecución (X, Execute)

- ocurre en todas las instrucciones
- Se ejecuta la operación en la ALU

Fase 4 -Acceso a memoria (M, Memory Access)

- ocurre solo en las instrucciones de LOAD y STORE
- Se accede a memoria

Fase 5 - Almacenamiento en Registro (W, Writeback)

- ocurre en las instrucciones que almacenan un dato en un registro
- También es en esta fase cuando puede calcular el desplazamiento a sumar al PC en instrucciones de salto.

NanoMIPS

Diagrama de estados

En la imagen anterior se pueden apreciar las 5 fase de ejecución:

Instrucción LOAD:

- la secuencia de fases es F-D-X-M-W(indicado como 0-1-2-3-4)
- Requiere las 5 fases

Instrucción STORE:

- la secuencia de fases es F-D-X-M (indicado 0-1-5-6)
- Usa 4 fases, no tiene fase W

Instrucción Aritmético/lógica:

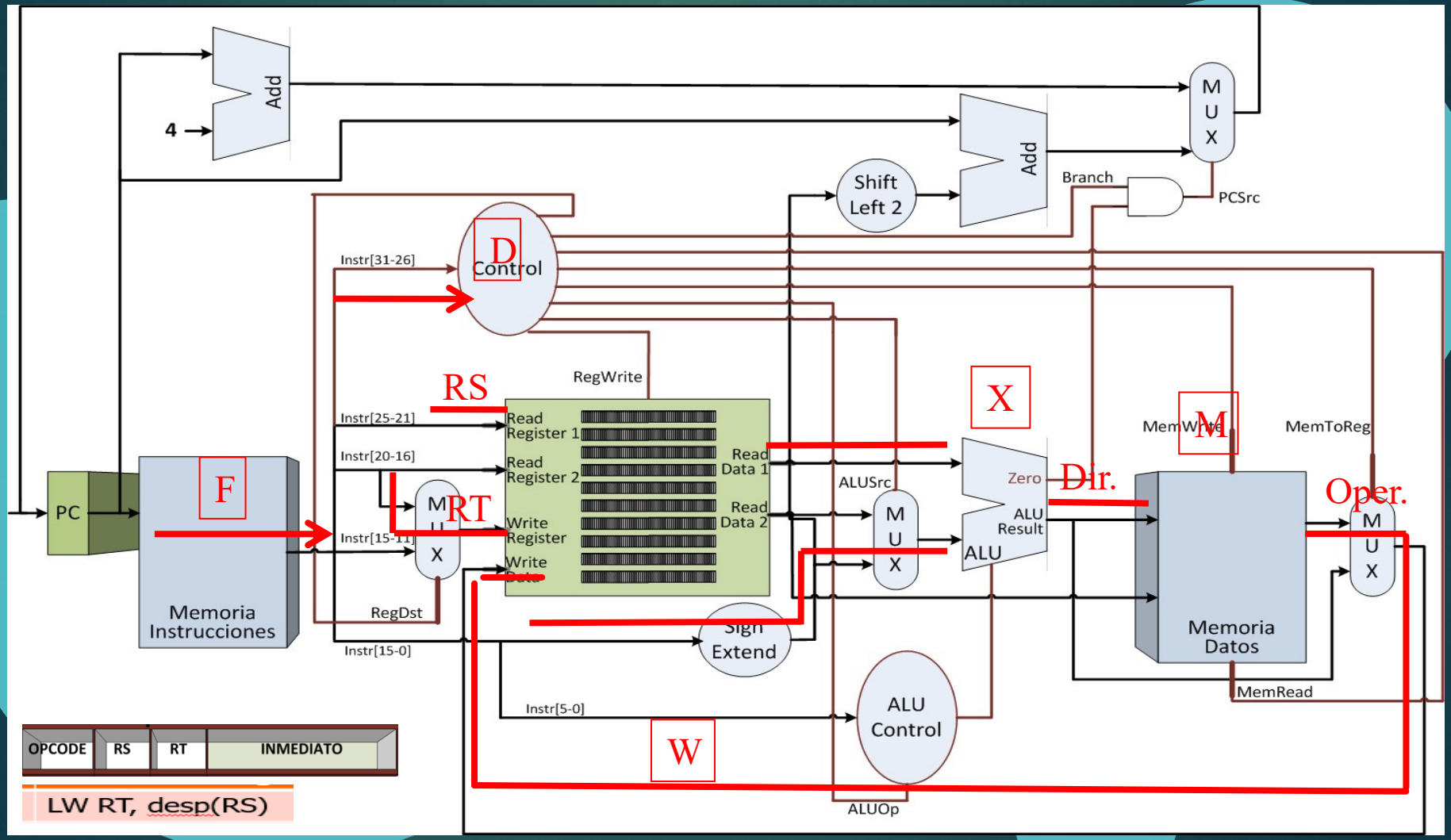
- la secuencia de fases es F-D-X-W (indicado 0-1-7-8)
- Usa 4 fases, no tiene fase M

Instrucción de salto:

- la secuencia de fases es F-D-X-W (indicado 0-1-9-10)
- Usa 4 fases, no tiene fase M

NanoMIPS

Secuencia de estados para una instrucción de LOAD



NanoMIPS

Diagrama de estados

En la imagen anterior se puede ver el proceso de ejecución de una instrucción de LOAD.

Fase F: Lee la instrucción.

- Los bits 26 a 31 son leídos por la Unidad de Control.
- Los bits 21 a 25 identifican el registro RS (registro puntero)
- Los bits 16 a 20 identifican el registro RT (destino o de escritura)
- Los bits 0 a 15 son el desplazamiento a sumar al RS

Fase D: la UC decodifica la instrucción y se accede al registro RS.

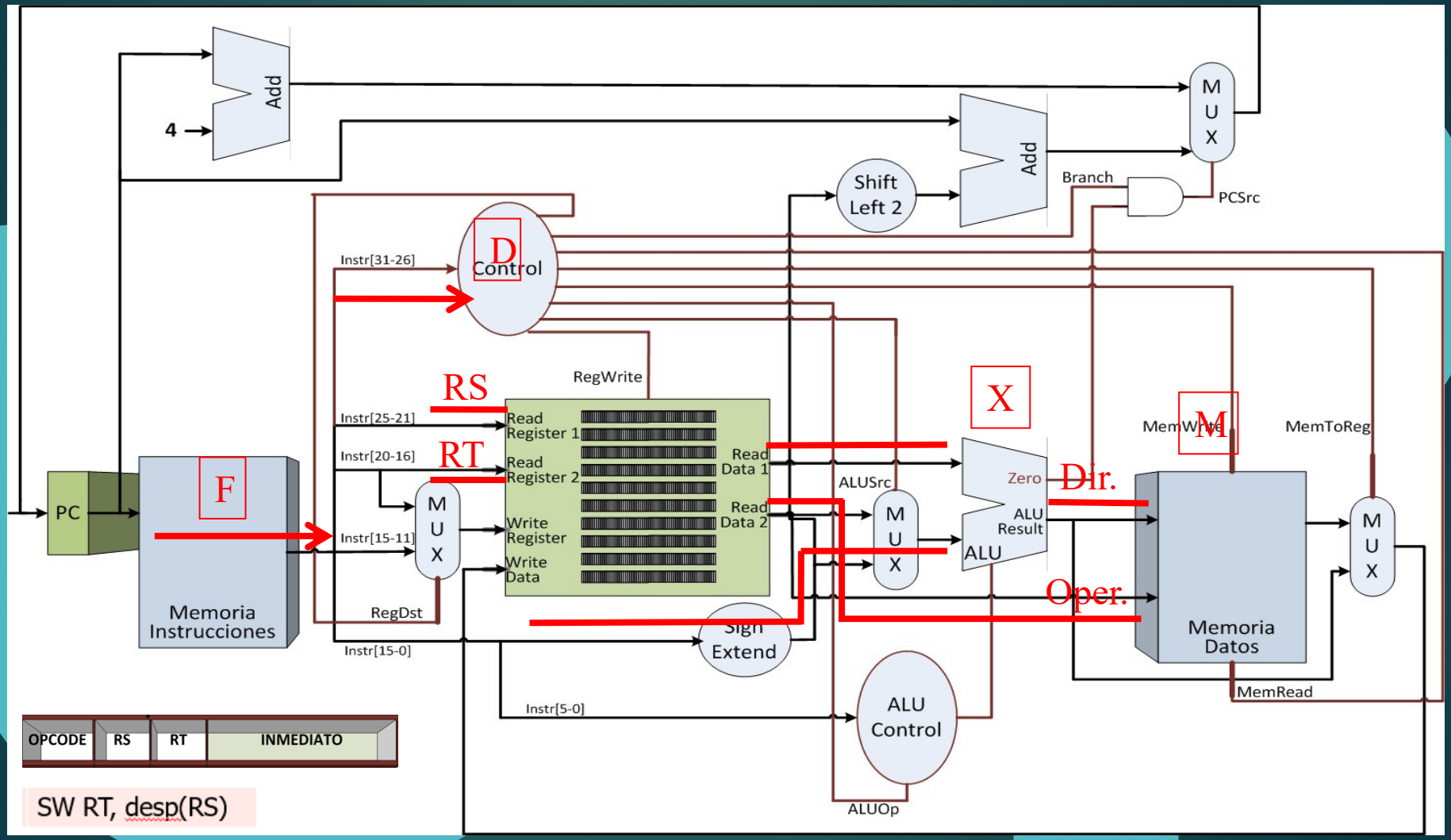
Fase X: se calcula la dirección efectiva donde está el operando a cargar en RT, como el valor de RS más el desplazamiento (inmediato).

Fase M: con la dirección efectiva, se accede a la memoria de datos, y se lee el dato a cargar en RT.

Fase W: el dato de la memoria se carga en el registro identificado con el campo RT de la instrucción.

NanoMIPS

Secuencia de estados para una instrucción de STORE



NanoMIPS

Diagrama de estados

En la imagen anterior se puede ver el proceso de ejecución de una instrucción de STORE.

Fase F: Lee la instrucción.

- Los bits 26 a 31 son leídos por la Unidad de Control.
- Los bits 21 a 25 identifican el registro RS (registro puntero)
- Los bits 16 a 20 identifican el registro RT (destino o de escritura)
- Los bits 0 a 15 son el desplazamiento a sumar al RS

Fase D: la UC decodifica la instrucción y se accede al registro RS.

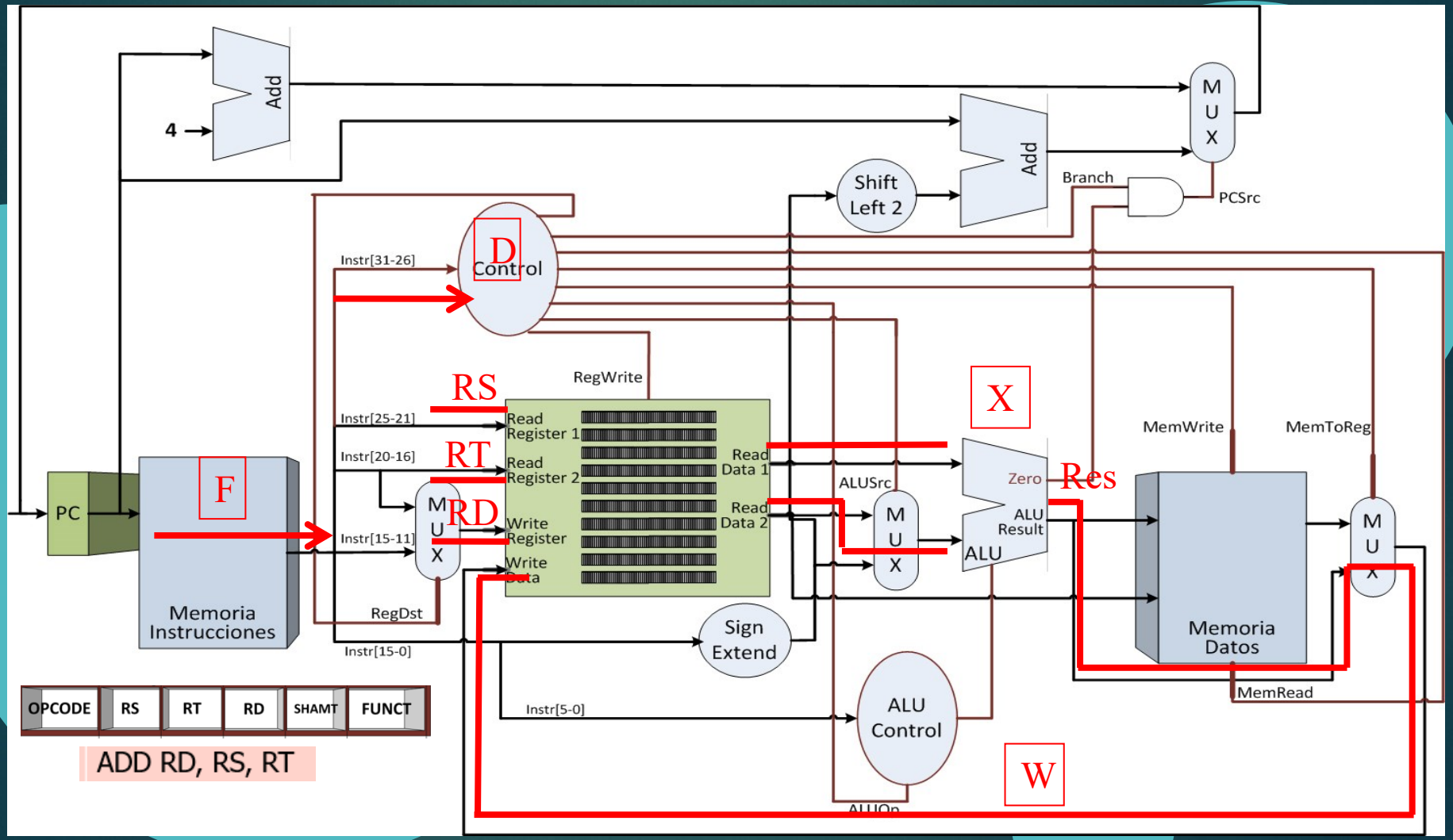
Fase X: se calcula la dirección efectiva, donde se va a almacenar el operando cargado en RT, como el valor de RS más el desplazamiento (inmediato).

Fase M: con la dirección efectiva, se accede a la memoria de datos, y se carga el dato almacenado en RT.

Fase W: no hay.

NanoMIPS

Secuencia de estados para una instrucción de ADD



NanoMIPS

Diagrama de estados

En la imagen anterior se puede ver el proceso de ejecución de una instrucción de ADD.

Fase F: Lee la instrucción.

- Los bits 26 a 31 son leídos por la Unidad de Control.
- Los bits 21 a 25 identifican el registro RS (1er operando)
- Los bits 16 a 20 identifican el registro RT (2do operando)
- Los bits 11 a 15 identifican el registro de destino RD

Fase D: la UC decodifica la instrucción y se accede a los registros RS y RT.

Fase X: se suman los contenidos de los registros RS y RT

Fase M: no hay

Fase W: se escribe el resultado en RD.

NanoMIPS

Ciclo de instrucción – Ejecución MONOCICLO

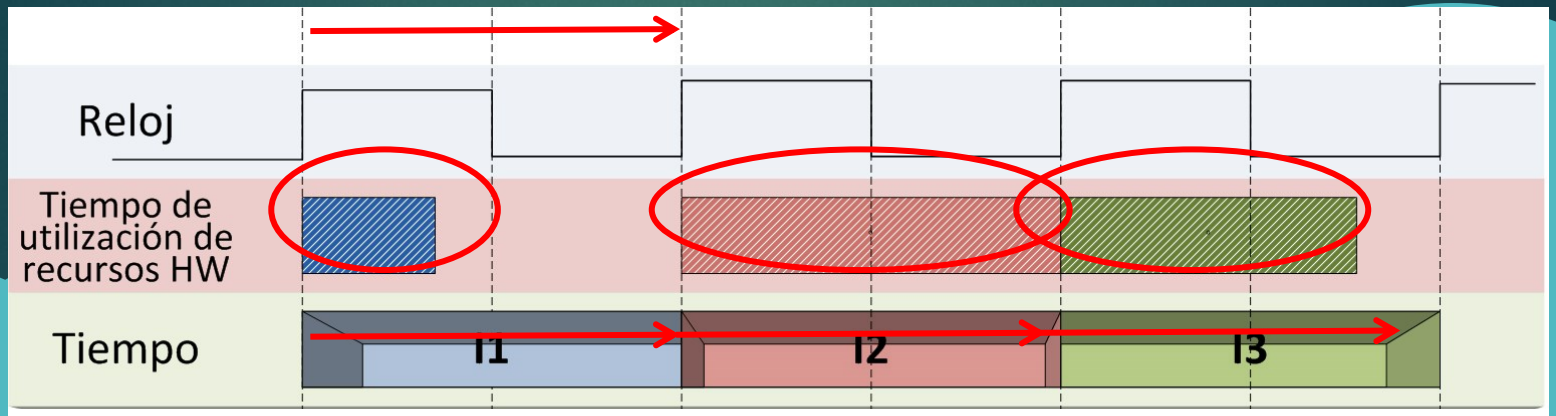
En las descripciones anteriores, cada vez que el PC envía una dirección a la memoria de instrucciones, se accede a una nueva instrucción.

Como no hay registros sincrónicos intercalados en las trayectorias de datos e instrucciones, el dato fluye por las unidades funcionales (registros, ALU, memoria, MUX, etc.) hasta que se complete la ejecución de la instrucción.

No se inicia un nuevo ciclo de instrucción hasta que el PC se carga con un nuevo valor (instrucción consecutiva o la de salto). Si el PC está sincronizado con un reloj (el de la CPU) significa que en cada ciclo de reloj se inicia una nueva instrucción. Este tipo de forma de operación se llama Ejecución MONOCICLO, y se muestra en la imagen siguiente.

NanoMIPS

nanoMIPS con ejecución Monociclo



- El período del reloj se hace con el tiempo necesario para completar la instrucción más lenta.
- En las instrucciones más rápidas el procesador “espera” el fin del ciclo para continuar (Ej: I1, I3).
- La eficiencia depende del tiempo de resolución de la instrucción más lenta (I2).

NanoMIPS

Ciclo de instrucción – Ejecución MULTICICLO

En la ejecución MONOCICLO el ciclo de reloj se hace en función de la instrucción “mas larga”, es decir la instrucción de LOAD.

Las instrucciones que son “mas cortas” tienen que esperar un tiempo hasta que se complete el período de reloj.

Si la tasa de instrucciones cortas es mucho mayor que la de las largas, se puede perder mucho tiempo inútilmente.

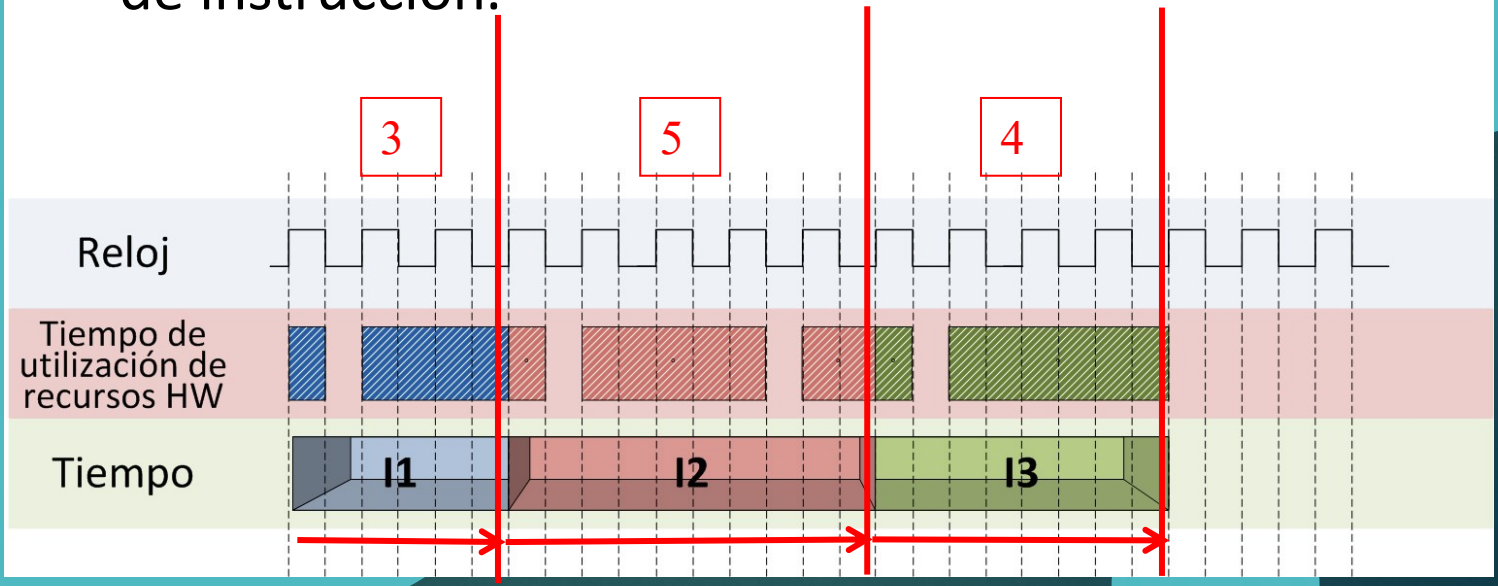
Una solución es fijar un ciclo de reloj más pequeño, y disponer que cada instrucción ocupe varios períodos de reloj.

Este tipo de forma de operación se llama Ejecución MULTICICLO, y se muestra en la imagen siguiente.

NanoMIPS

Ciclo de instrucción – Ejecución MULTICICLO

- En la ejecución MULTICICLO cada instrucción requiere varios ciclos de reloj.
- Las instrucciones más rápidas usan menos ciclos de reloj que las más lentas.
- El ciclo de instrucción es variable y se ajusta al tipo de instrucción.



Segmentación y solapamiento del cauce (Pipelining)

La segmentación del cauce (Pipelining) consiste en:

- Descomponer el proceso de ejecución de las instrucciones en fases o etapas.
- Las fases o etapas son ejecutadas por unidades separadas y capaces de operar simultáneamente.
- Las instrucciones se van ejecutando a medida que se liberan las unidades.
- Las instrucciones no necesitan esperar la terminación de la previa para comenzar a resolverse.
- La segmentación del cauce (pipelining) es una forma particularmente efectiva de organizar el hardware de la CPU para realizar más de una tarea al mismo tiempo. Explota el paralelismo en el flujo secuencial de instrucciones.

Segmentación y solapamiento del cauce (Pipelining)

Para entender el concepto de segmentación de una tarea repetitiva se puede analizar un "proceso" de una lavandería que requiere 3 etapas:

- Lavado, que requiere 30 minutos
- Secado, que requiere 40 minutos
- Despacho, que requiere 20 minutos

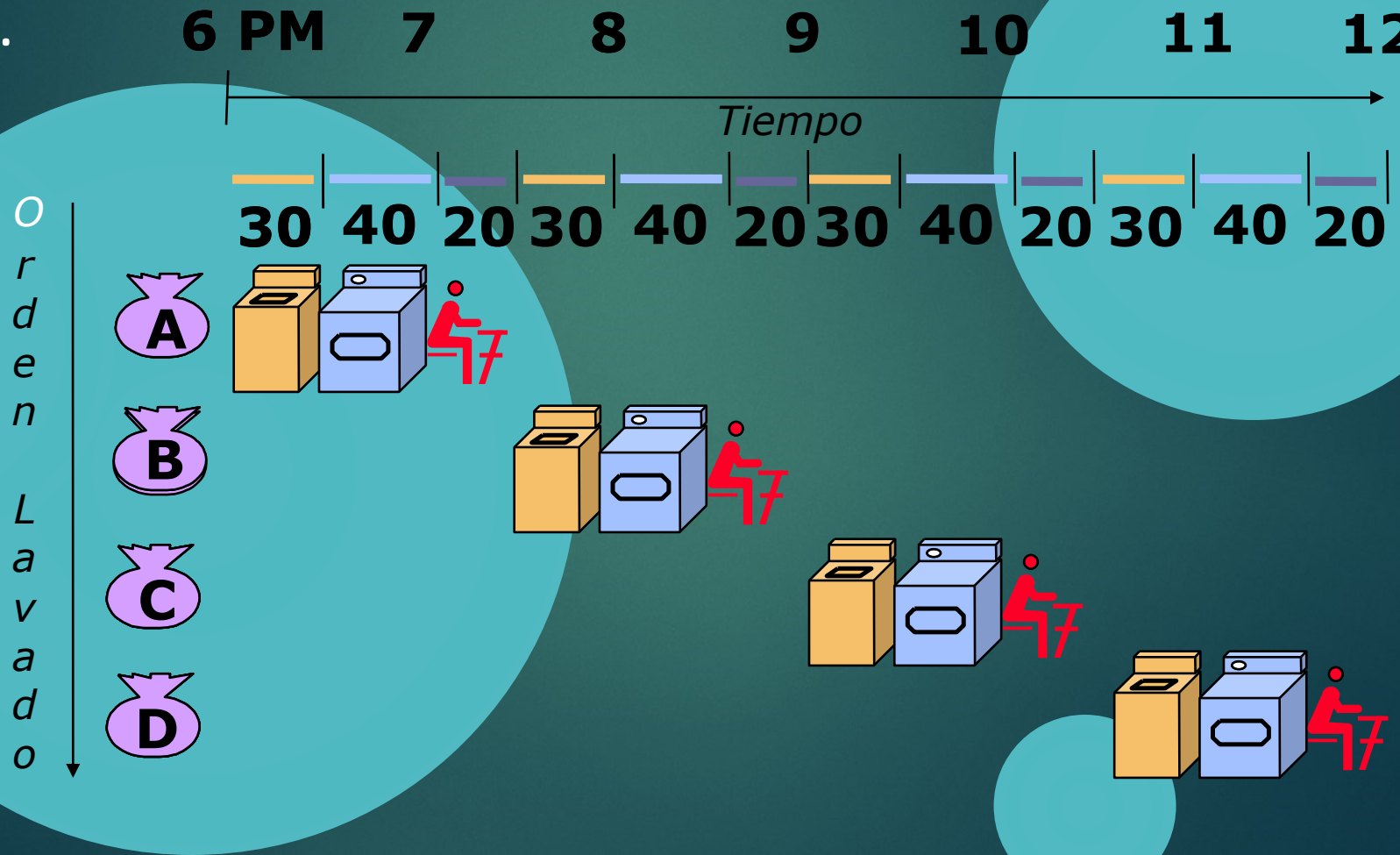
Las 3 etapas se ejecutan en 3 sectores (estaciones) separados.

La primera forma de operar es ejecutar el proceso en forma puramente secuencial, cuando se termina con un encargue recién se empieza con el siguiente.

Este modelo secuencial se muestra en la figura siguiente.

Segmentación y solapamiento del cauce (Pipelining)

Proceso con ejecución "secuencial": cada tarea (A, B, C, D) require en total 90 minutos (30+40+20). Tiempo total = 6 horas.



Segmentación y solapamiento del cauce (Pipelining)

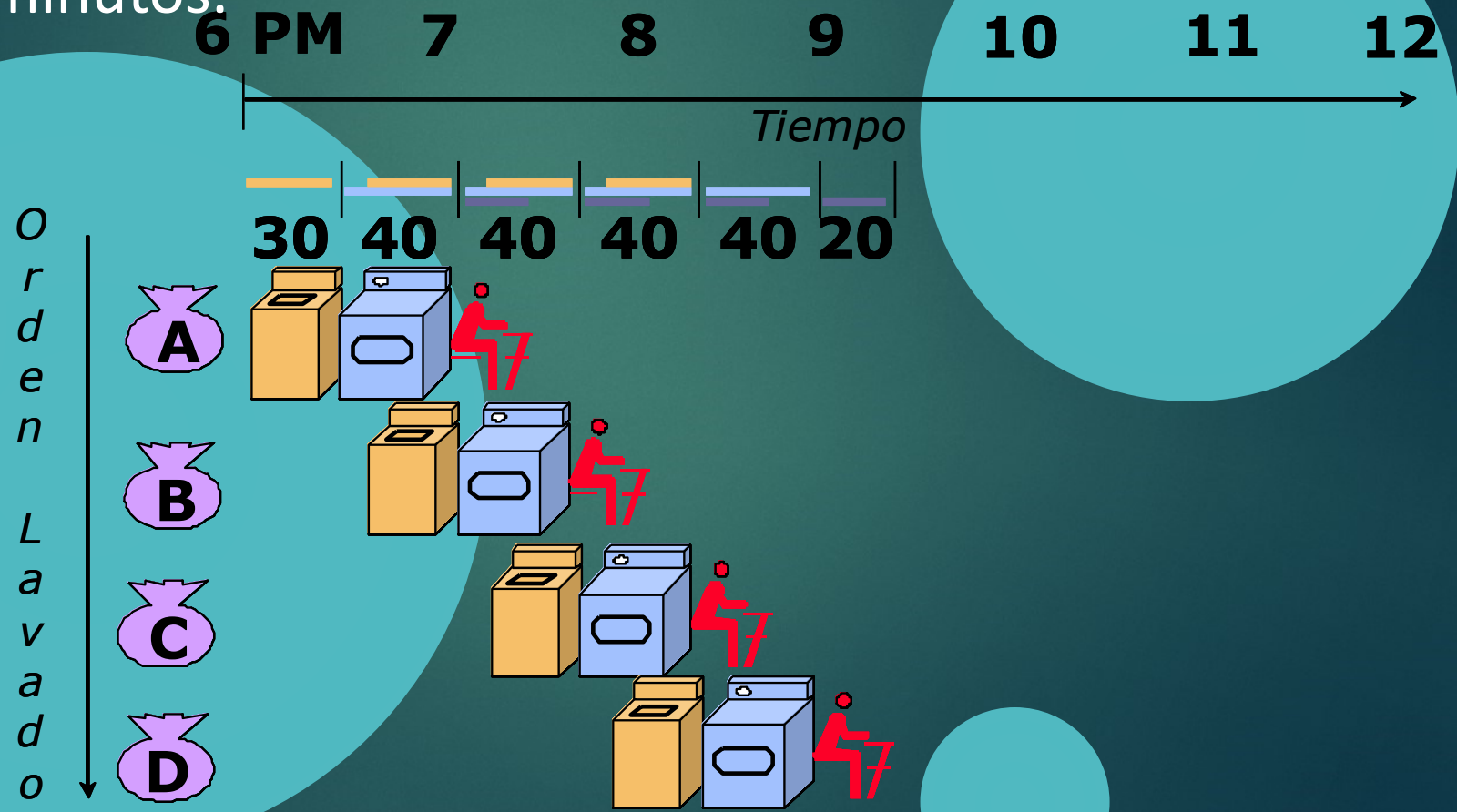
Como las unidades funcionales Lavado, Secado y Despacho son unidades distintas, pueden operar en forma simultánea.

Si se superponen (solapan) las tareas de las 3 unidades funcionales es posible obtener una mejora sustancial en el tiempo total de ejecución de los 4 encargues (A, B, C y D).

Esa situación se muestra en la figura siguiente.

Segmentación y solapamiento del cauce (Pipelining)

Proceso con ejecución "solapada": cada tarea (A, B, C, D) require en total 90 minutos (30+40+20). Tiempo total = 3 horas 30 minutos.



Segmentación y solapamiento del cauce (Pipelining)

- En el ejemplo anterior se puede observer que cada tarea lleva exactamente el mismo tiempo: 30+40+20 = 90 minutos.
- La mejora se obtiene debido al solapamiento de las tareas de las distintas unidades funcionales.
- Solapando las tareas A, B, C y D, se consigue realizarlas en 3horas 30 minutos, en lugar de las 6 horas que se tardaba operándolas en forma secuencial.
- Se puede aplicar este concepto al proceso repetitivo de ejecutar una secuencia de instrucciones.

Segmentación y solapamiento del cauce (Pipelining)

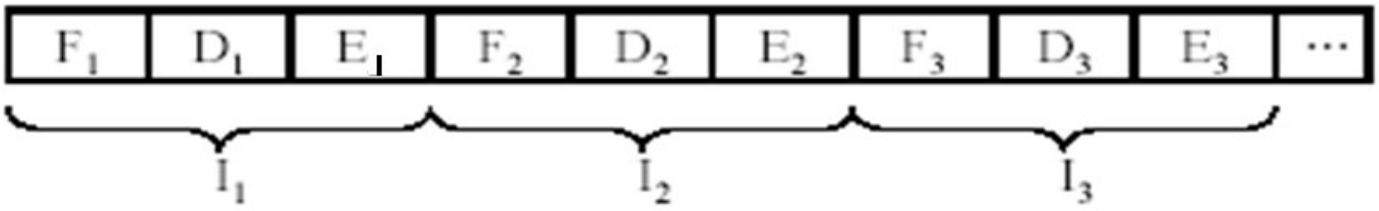
Supongamos que se tiene una máquina que resuelve las instrucciones en 3 fases:

- Búsqueda o Captación (F, Fetch)
  - Acceso a memoria (búsqueda de la instrucción)
  - Incremento del PC
- Decodificación (D, Decode)
  - Decodificación de la instrucción
  - Obtención de los operandos
- Ejecución (E, Execute)
  - Si es procesamiento: ejecución en la ALU
  - Si es acceso a memoria: obtención de la direc efectiva
  - Si es salto: cálculo del destino y decisión de salto (s/n)

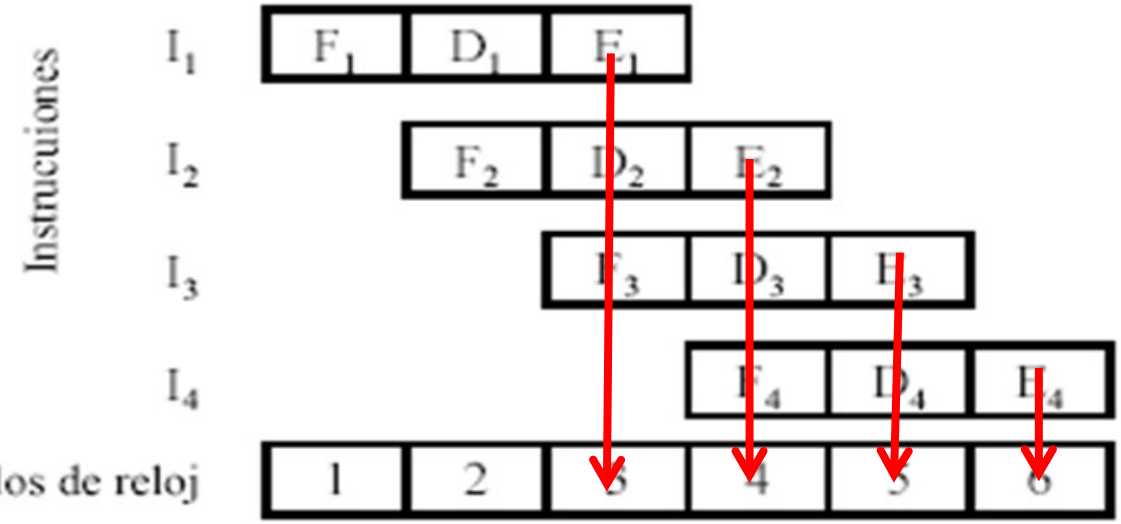
Segmentación y solapamiento del cauce (Pipelining)

La secuencia de tareas se puede hacer secuencialmente o segmentadamente.

- Ejecución secuencial:



- Ejecución segmentada (tres etapas)



Segmentación y solapamiento del cauce (Pipelining)

- En el procesador con ejecución secuencial, se tardan 9 ciclos de reloj para completer 3 instrucciones.
- En el procesador segmentado de la figura anterior, de 3 etapas, se tardan 5 ciclos de reloj para completar las mismas 3 instrucciones.
- En ambos casos, cada instrucción require 3 ciclos de reloj.

Segmentación y solapamiento del cauce (Pipelining)

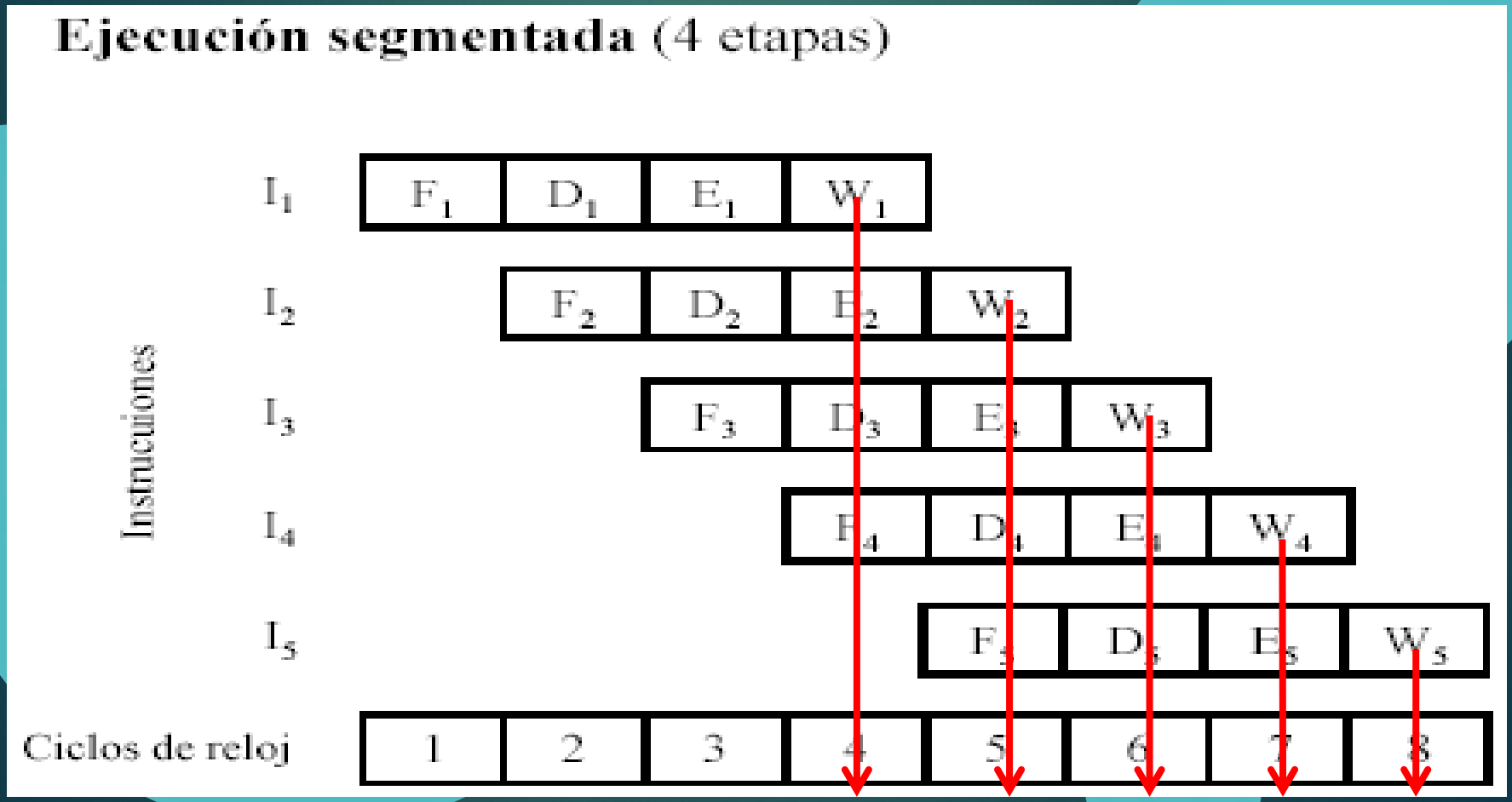
Si la máquina tiene 4 fases para resolver una instrucción:

- Búsqueda o Captación (F, Fetch)
- Decodificación (D)
- Ejecución (E)
- Escritura o actualización (W)

La segmentación queda como se muestra en la figura siguiente.

Segmentación y solapamiento del cauce (Pipelining)

Ejecución segmentada en una máquina de 4 etapas F, D, E, W, de una secuencia de 5 instrucciones.



Segmentación y solapamiento del cauce (Pipelining)

- En el procesador segmentado de 4 etapas de la figura anterior, se tardan 8 ciclos de reloj para completar 5 instrucciones.
- En el procesador con ejecución secuencial se tardarían 5x4=20 ciclos de reloj para completer las 5 instrucciones.
- En ambos casos, cada instrucción require 4 ciclos de reloj.

Segmentación y solapamiento del cauce (Pipelining)

Conclusiones:

- El comportamiento de un procesador ejecutando una secuencia de instrucciones se comporta de una forma similar a una línea de montaje en una planta de manufactura.
- Cada instrucción pasa, durante su ejecución, por varias etapas, en cada una se realiza solo una parte del todo.
- La entrada de una nueva instrucción puede hacerse antes de que se terminen las anteriores.
- Por lo tanto, varias instrucciones están siendo manipuladas simultáneamente, cada una en un estado de ejecución distinto.

Segmentación y solapamiento del cauce (Pipelining)

Conclusiones:

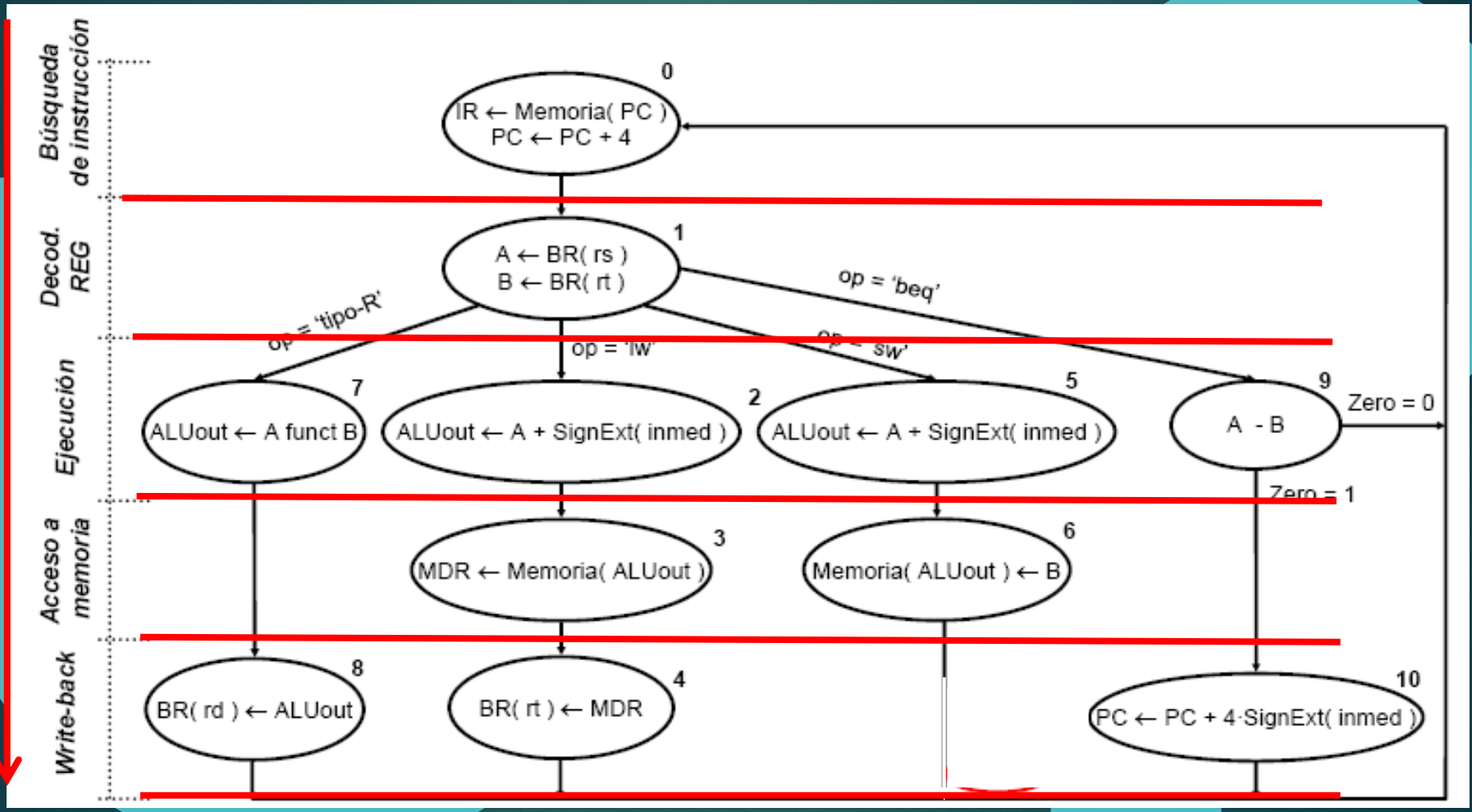
- Dado que el procesador tarda menos tiempo en resolver un conjunto de instrucciones, la segmentación mejora las prestaciones (a nivel de diseño del hardware).
- Como el programador no interviene en la segmentación y solapamiento de las instrucciones, la segmentación es "invisible" (por ahora) al programador.
- Se puede usar tanto en procesadores RISC como CISC.
- El diseño de procesadores segmentados tiene gran dependencia del repertorio de instrucciones.

Segmentación y solapamiento del cauce en el nanoMIPS

- Se puede aplicar el concepto de Segmentación en el nanoMIPS.
- De acuerdo a lo visto, el proceso de ejecución de las instrucciones en el nanoMIPS está compuesto por 5 fases o etapas.
- En la imagen siguiente se puede apreciar la segmentación de las fases de ejecución de una instrucción.

Segmentación y solapamiento del cauce en el nanoMIPS

Ejecución segmentada en 5 etapas en el nanoMIPS



Segmentación y solapamiento del cauce en el nanoMIPS

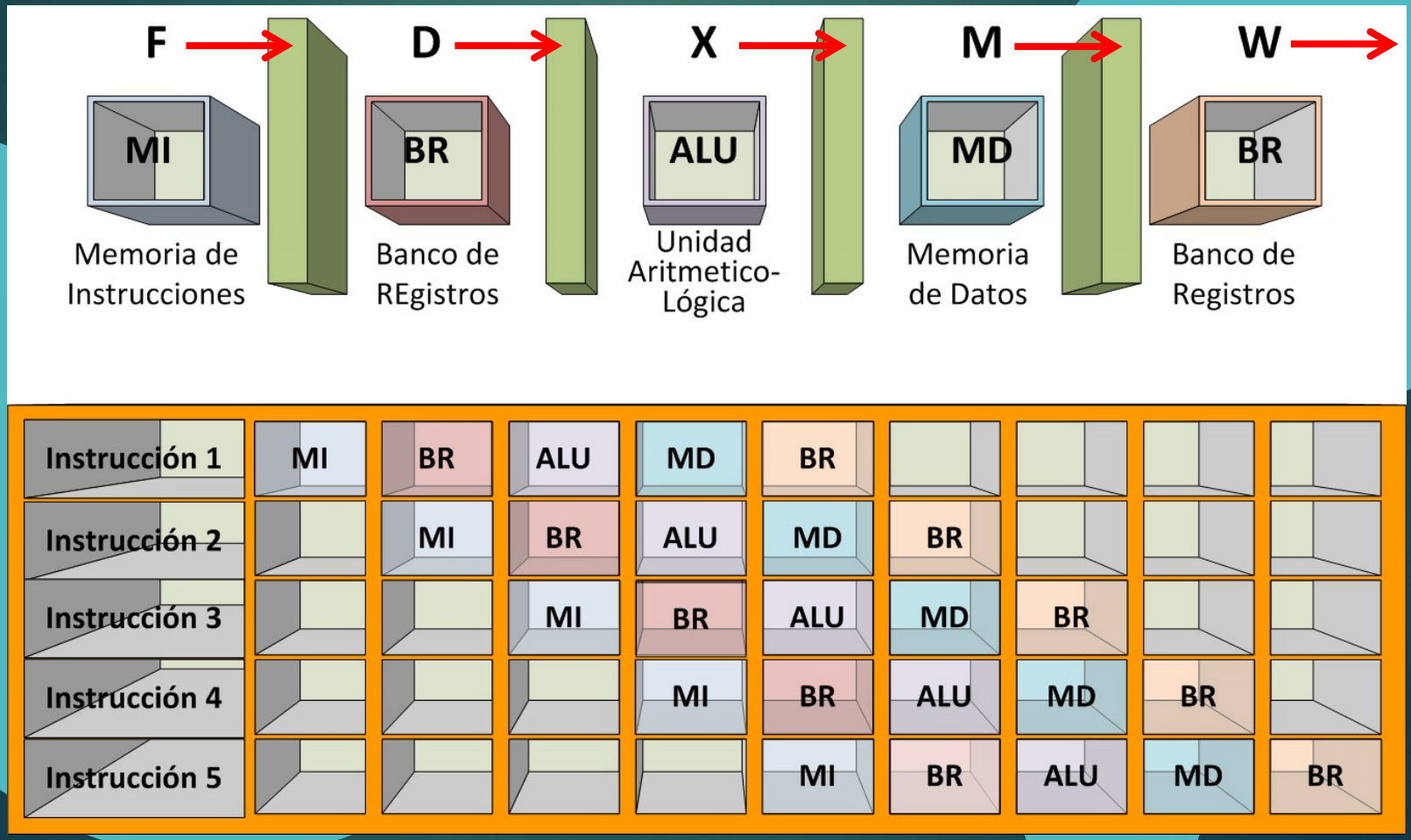
- Los 5 segmentos del nanoMIPS son:
  - F: búsqueda de la insttrucción en la MI
  - D: decodificación y acceso a registros en el BR
  - X: ejecución de la ALU
  - M: acceso a la memoria de datos MD
  - W: escritura del resultado en el banco de registros BR
- Y los recursos de la máquina son:
  - MI: memoria de instrucciones
  - BR: banco de registros
  - ALU: ALU
  - MD: memoria de datos

Segmentación y solapamiento del cauce en el nanoMIPS

- Para poder aplicar el concepto de segmentación en el nanoMIPS, hay que dividir (segmentar) el cauce de los datos en etapas.
- La forma de hacer esto es usando registros sincrónicos entre cada etapa.
- Los datos avanzan, de etapa a etapa, en cada ciclo de reloj, que es cuando el registro (de separación de la etapa) copia a la salida lo que tiene en su entrada.
- Los segmentos en que queda dividido el cauce de datos, corresponden a las 5 fases de ejecución de la instrucción: F, D, X, M y W.
- Este proceso se puede apreciar en la siguiente figura.

Segmentación y solapamiento del cauce en el nanoMIPS

Ejecución segmentada en 5 etapas en el nanoMIPS, de un secuencia de 5 instrucciones.



Segmentación y solapamiento del cauce en el nanoMIPS

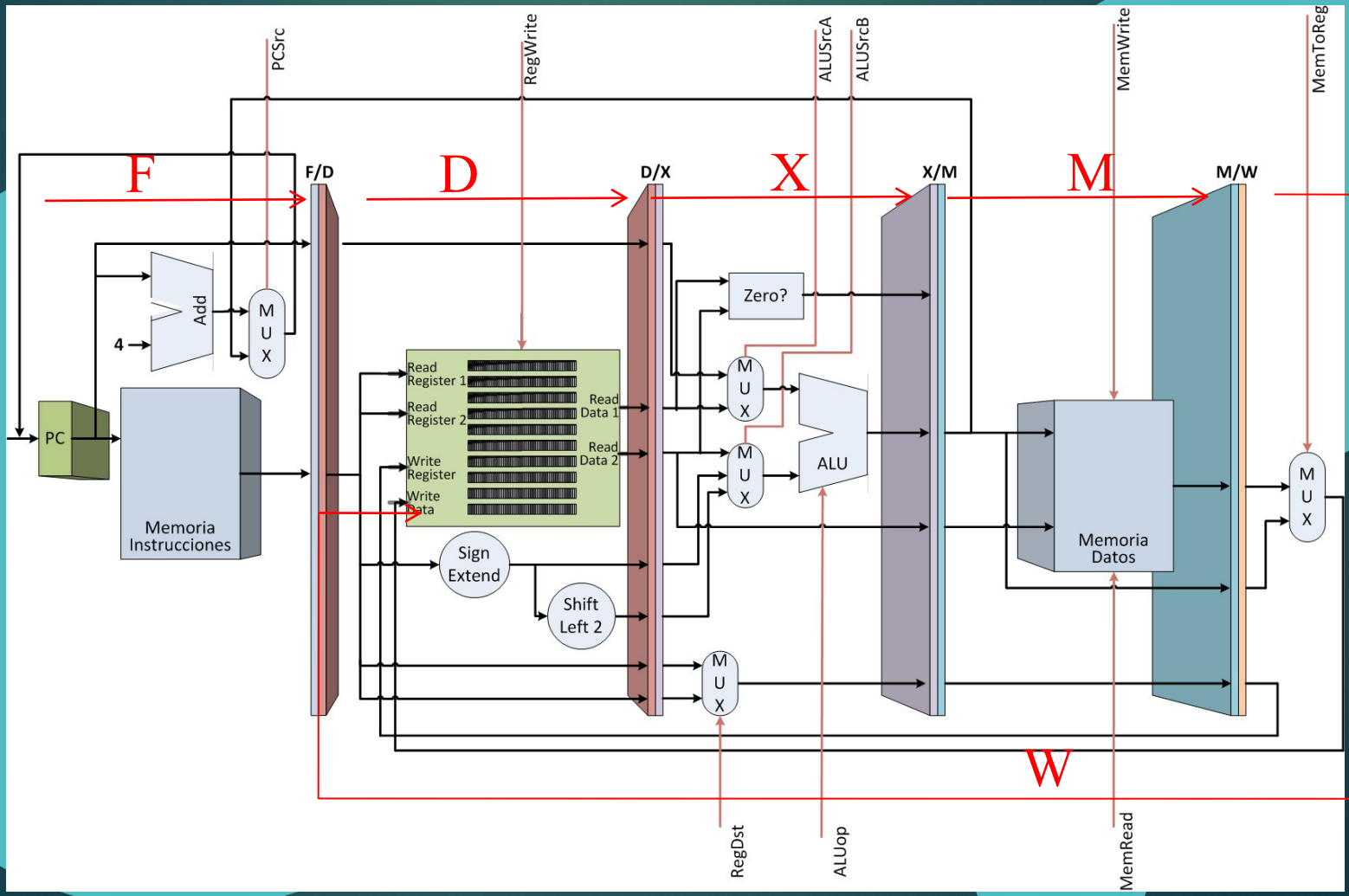
- En la figura anterior se puede apreciar el proceso de ejecución de una secuencia de 5 instrucciones.
- Se considera que cada instrucción require 5 ciclos de reloj, uno por cada segmento (F-D-X-M-W).
- En cada ciclo se incorpora una nueva instrucción, sin haber completado las anteriores.
- La primera instrucción se termina en el período de reloj 5, hasta ese ciclo no se había resuelto ninguna instrucción.
- A partir del quinto ciclo, se termina 1 instrucción por ciclo.
- Notar que en el quinto ciclo el procesador está ejecutando 5 instrucciones simultáneamente, pero en distintas fases.

Segmentación y solapamiento del cauce en el nanoMIPS

- Para implementar la segmentación del cauce se requiere intercalar registros entre cada etapa.
- Los registros (sincrónicos) separan las unidades funcionales, para que puedan operar con distintas instrucciones.
- En cada ciclo de reloj el dato avanza de una unidad funcional a la siguiente en el cauce, hasta completar las fases requeridas por a instrucción.
- El cauce de datos queda segmentado de la siguiente manera.

Segmentación y solapamiento del cauce en el nanoMIPS

Segmentación del cauce de datos.

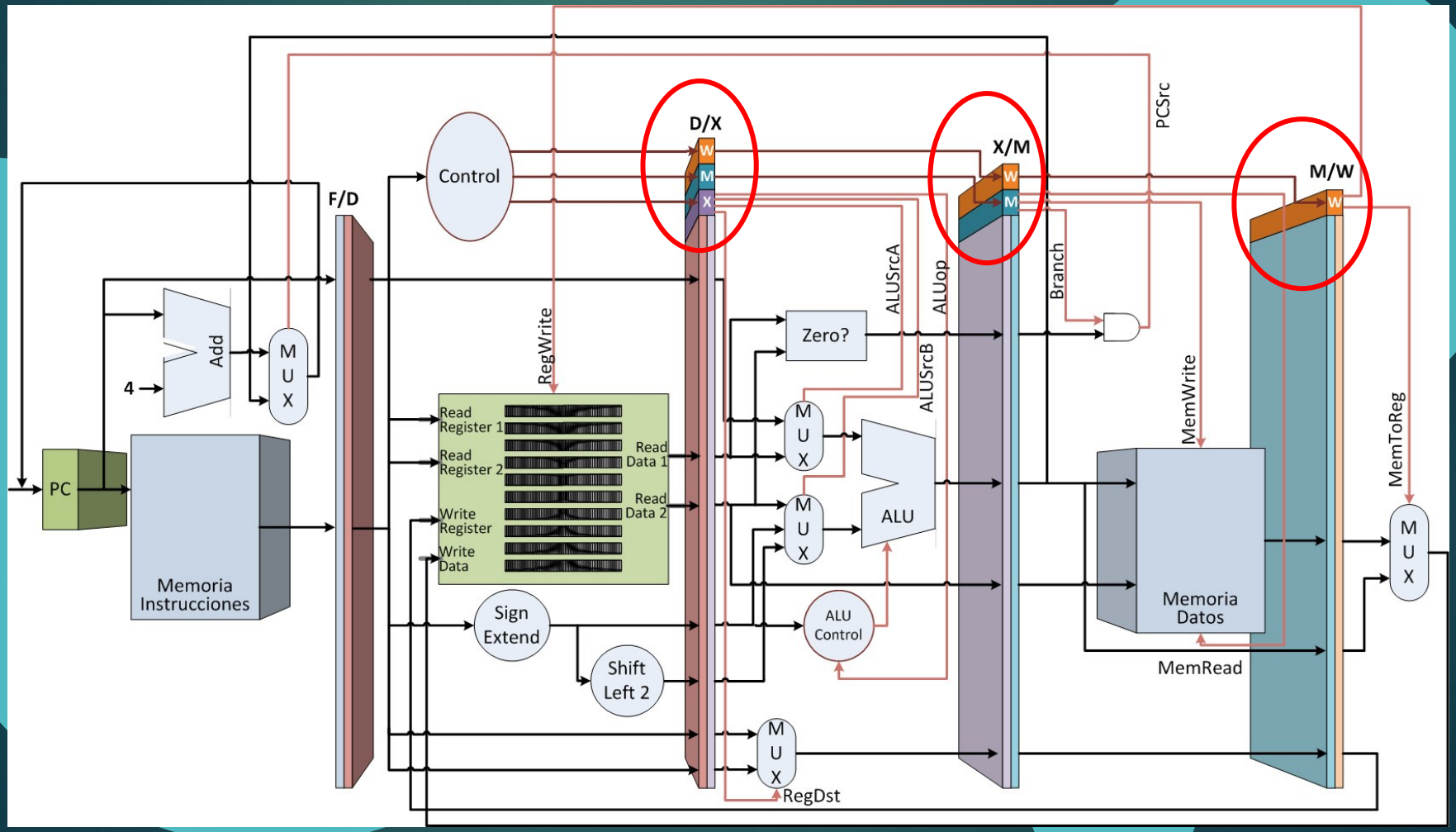


Segmentación y solapamiento del cauce en el nanoMIPS

- Así como se segmenta el cauce de los datos, también se debe hacer lo mismo con las señales que controlan las unidades funcionales: MI, BR, ALU, MD, BR.
- Eso es así porque cada Unidad funcional está operando con diferentes instrucciones (en diferentes estados de ejecución).
- Por ejemplo, mientras la unidad funcional BR (Banco de registros) está buscando los operandos de la instrucción I1, la unidad funcional MI (Memoria de instrucciones) estará buscando la instrucción I2.
- Las señales de control se segmentan igual que el cauce de datos, usando registros sincrónicos entre cada etapa, como se muestra en la figura siguiente.

Segmentación y solapamiento del cauce en el nanoMIPS

Segmentación del cauce de datos y de control.



Análisis de la Segmentación

- El máximo rendimiento teórico se obtiene cuando se completa una instrucción en cada ciclo de reloj. En esas condiciones todas las unidades funcionales están trabajando simultáneamente con distintas instrucciones.
- Si K es el número de etapas del cauce, entonces:

  Vel. procesador segmentado = K x Vel. secuencial

- El incremento potencial de la segmentación del cauce es proporcional al número de etapas del cauce.
- Notar que no se mejora la velocidad de ejecución de la instrucción, la segmentación incrementa la productividad (throughput), es decir la cantidad de instrucciones resueltas en un período de tiempo determinado.

Análisis de la Segmentación

Para el análisis anterior se han hecho algunas suposiciones:

- Todas las tareas o segmentos duran la misma cantidad de ciclos de reloj (tiempo).
- Todas las instrucciones siempre pasan por todas las mismas etapas.
- Todos las etapas pueden ser manejadas en paralelo (no hay conflictos para usarlas simultáneamente).
- No se consideraron instrucciones de salto.

Para hacer un análisis más preciso se requiere estudiar el comportamiento de la segmentación, pero en condiciones mas realistas.

Análisis de la Segmentación

Las primeras correcciones a hacer son:

1.- No todas las instrucciones necesitan todas las etapas.

- Ej: en el nanoMips la instrucción SW RT, inmed(RS) no utiliza W
- Ej: en el MSX88: un MOV AX,mem ; no requiere X

2.- No todas las etapas pueden ser manejadas en paralelo.

- Ej: si la memoria no estuviera dividida y fuera una sola, los segmentos F (búsqueda de la instrucción) y M (acceso a memoria) accederían ambos a la memoria (única) de datos e instrucciones.

3.- Los programas tienen instrucciones de salto.

Análisis de la Segmentación

Los conflictos que aparecen si se tienen en cuenta las correcciones anteriores, se denominan riesgos.

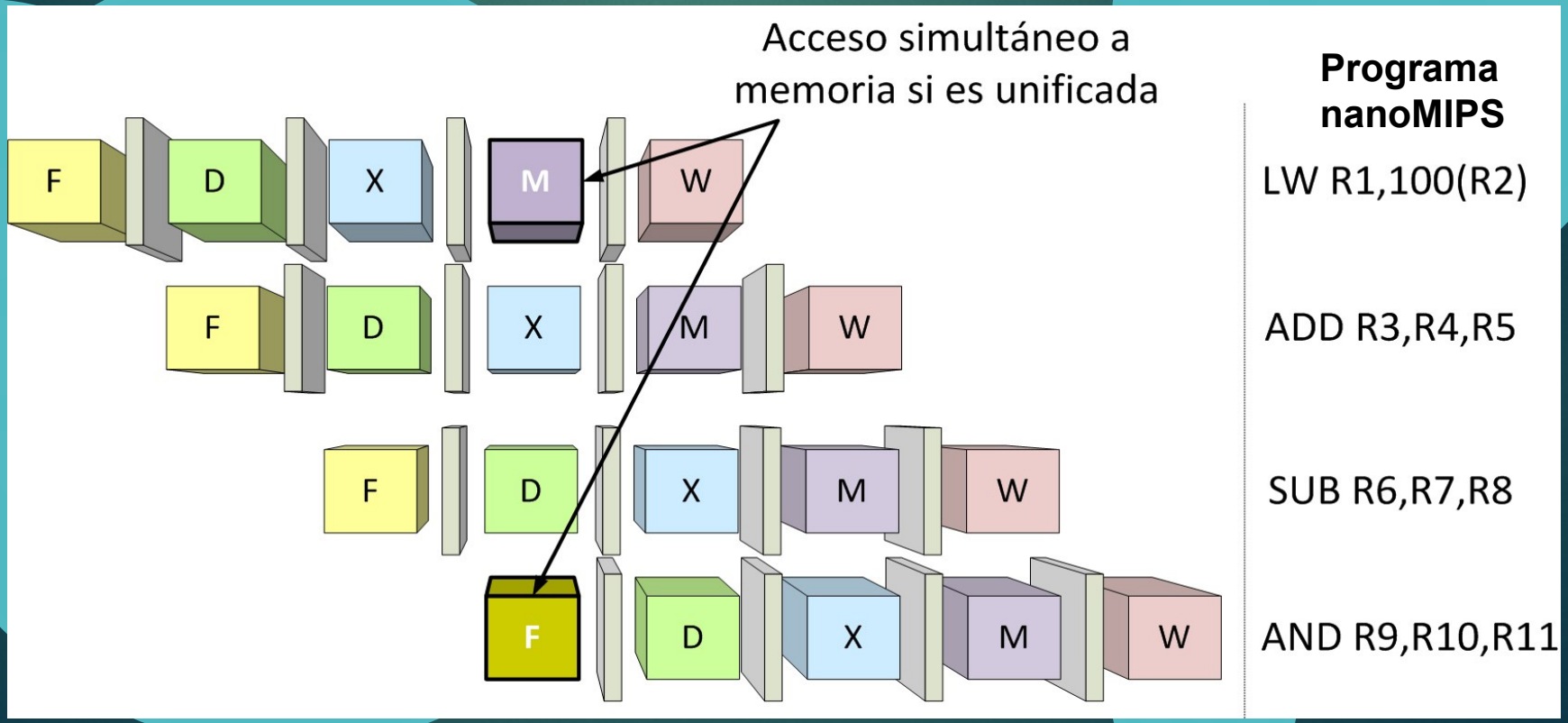
Existen 3 tipos de riesgos:

- 1.- Estructurales: son conflictos provocados por el uso de los "recursos". Los recursos típicamente son: memoria, ALU, registros.
- 2.- Por dependencia de datos: son conflictos originados entre 2 o más instrucciones que comparten un mismo dato. Por ejemplo, una instrucción produce un resultado que lo necesita otra, ambas instrucciones dentro del cauce de datos.
- 3.- Por dependencia de control: son conflictos que ocurren cuando la ejecución de una instrucción depende de cómo se ejecute otra (ej.: un salto y los 2 posibles caminos).

Análisis de la Segmentación

Ejemplo de conflicto por uso de un recurso: la Memoria

Si la memoria del nanoMIPS fuera una sola (instrucciones y datos) habría un conflicto cada vez que se solapan los ciclos M y F.



Análisis de la Segmentación

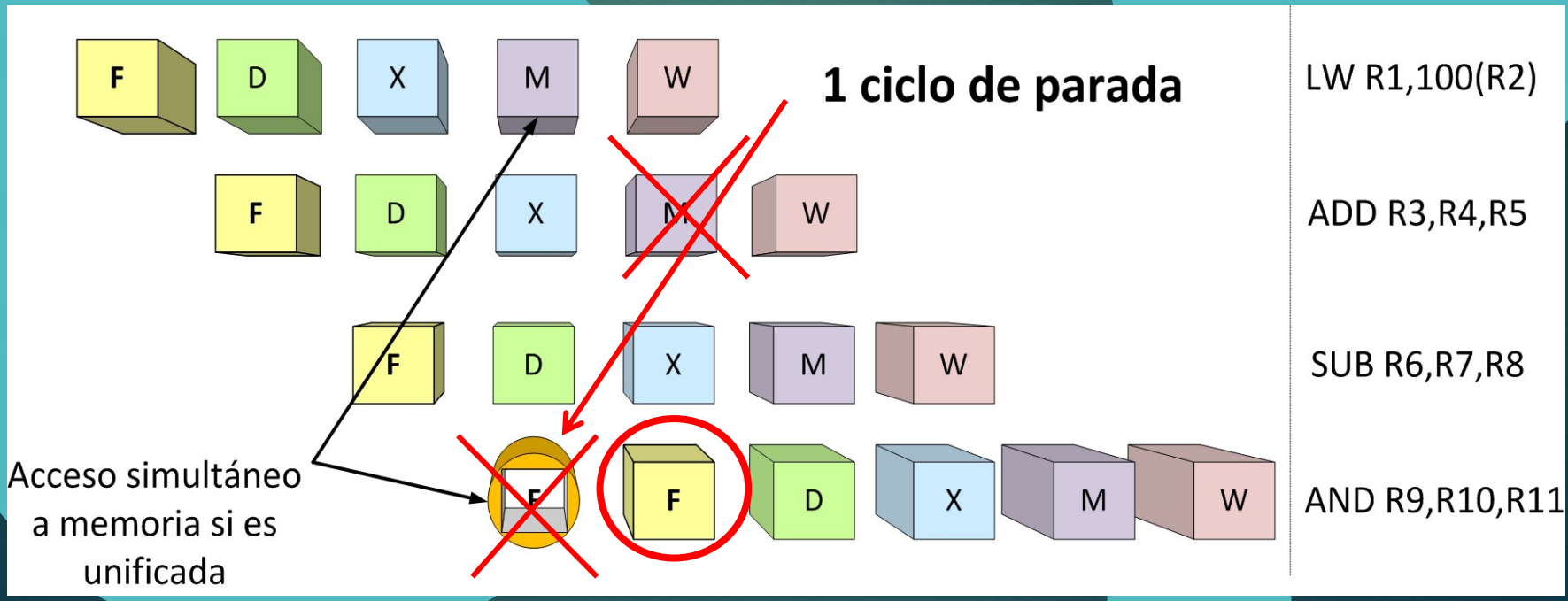
Riesgos Estructurales

- En la imagen anterior se muestra la situación de un conflicto entre las fases F y M si la memoria fuera una sola.
- Por esta razón, la memoria del nanoMIPS está dividida en MI y MD, de manera de reducir los conflictos por accesos a memoria.
- Otra opción para evitar el conflicto por el acceso a un recurso es retardar (retrasar) la ejecución de la tarea los ciclos de reloj necesarios hasta que desaparece el conflicto. Pero al perder ciclos de reloj se pierde tiempo y baja la performance del procesador. Y además, el retraso puede generar nuevos conflictos que también deben resolverse de alguna manera.
- Esta solución se muestra en la figura siguiente.

Análisis de la Segmentación

Ejemplo de resolución del conflicto por uso de un recurso: la Memoria

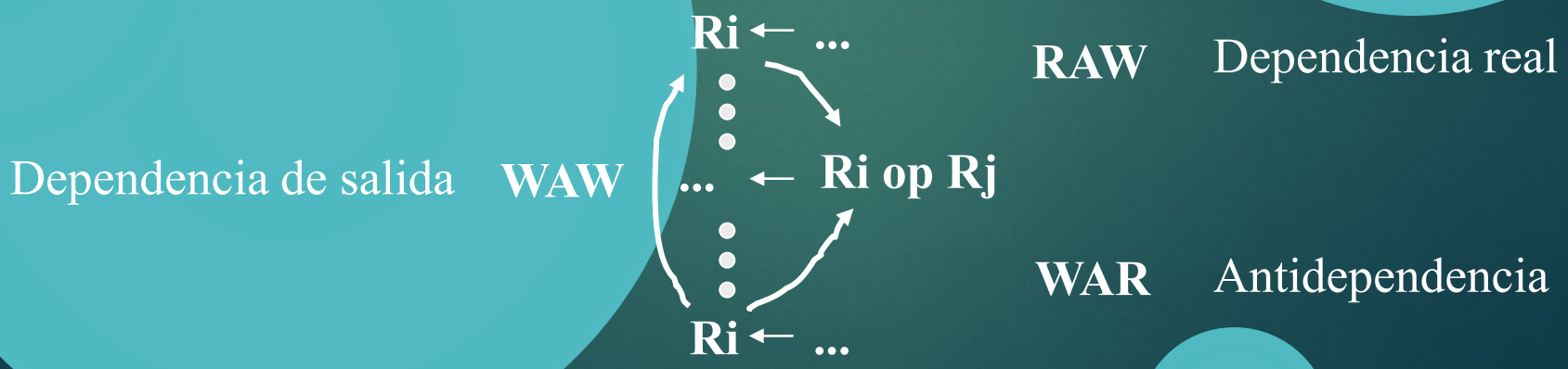
Se retrasa el inicio de la instrucción AND R9,R10,R11 en 1 ciclo (ciclo de parada) para evitar el conflicto con la instrucción LW R1,100(R2).



Análisis de la Segmentación

2.- Por dependencia de datos:

- Hay riesgos de conflictos por dependencia de datos, cuando un dato es usado en 2 o más segmentos del cauce.
- Los operandos fuente o destino de una instrucción no están disponibles en el momento en que se necesitan en una etapa determinada del cauce.
- Se pueden dar 3 tipos de dependencias:



Análisis de la Segmentación

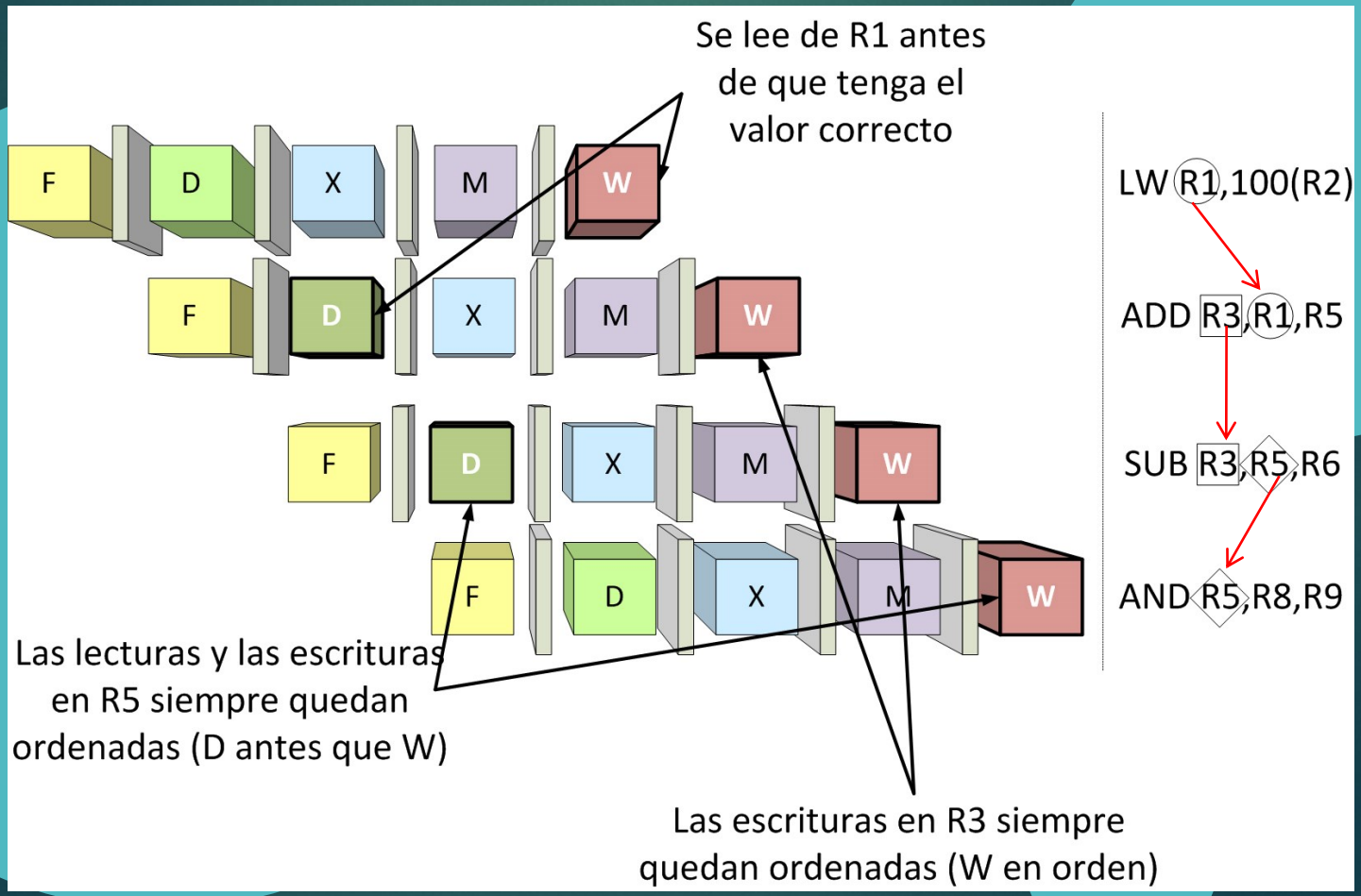
Los 3 tipos de dependencias de datos son:

- Lectura después de Escritura (RAW, dependencia verdadera)
  - Una instrucción escribe un dato que otra lee posteriormente
- Escritura después de Escritura (WAW, dependencia en salida)
  - una instrucción escribe un dato que otra escribe posteriormente
  - sólo ocurre si se permite que las instrucciones se adelanten unas a otras (alteración en la secuencia de ejecución de instrucciones)
- Escritura después de Lectura (WAR, antidependencia)
  - una instrucción lee un dato que otra escribe posteriormente
  - no se puede dar en nuestro cauce simple

Análisis de la Segmentación

Ejemplo de conflicto por dependencia de datos

Se muestra una secuencia de 4 instrucciones con los “3 tipos de dependencias”.

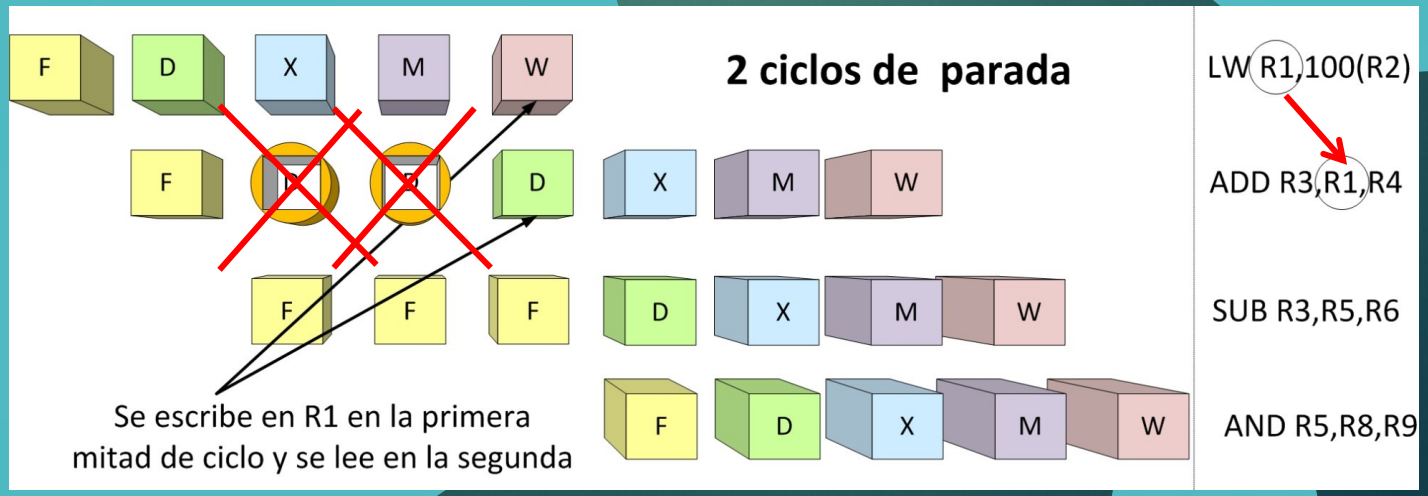


Análisis de la Segmentación

Ejemplo de resolución del conflicto por dependencia de datos

Se retrasa la decodificación D de la instrucción ADD R3,R1,R4 3 ciclos para esperar tener el dato disponible de la instrucción LW R1,100(R2).

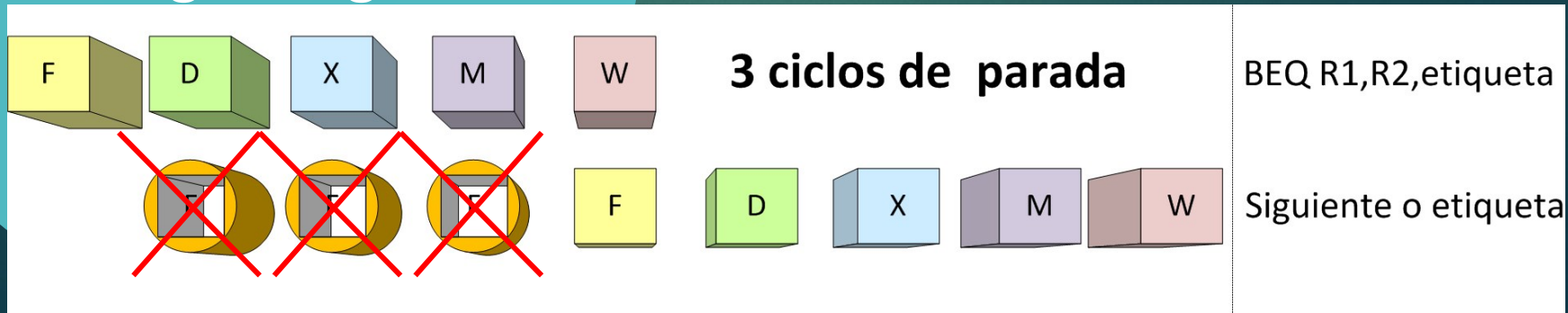
Se considera que la escritura W de la instrucción LW R1,100(R2) se hace en el primer medio ciclo y se puede leer por la instrucción ADD R3,R1,R4 en el segundo semiciclo.



Análisis de la Segmentación

3.- Por dependencia de control:

- Son riesgos que pueden ocurrir cuando se va a ejecutar una instrucción de salto condicional.
- Una instrucción tiene que calcular el nuevo valor que modifica el valor del PC. La próxima instrucción no puede comenzar hasta que no se resuelva el salto.
- Una forma de resolver el conflicto es retrasar varios ciclos (3 ciclos) la próxima instrucción hasta que se resuelva el cálculo de la instrucción de salto, como se puede apreciar en la figura siguiente.



Referencias

- W. Stallings, 5º Ed - Capítulo 11.
- Hwang & Briggs – Computer Architecture & Parallel Processing – Chapter 10.
- M. Pardo y A. Sacristán - Diseño y evaluación de arquitecturas de computadoras.