# Arquitectura de Computadoras

**Clase 4**

**Segmentación de Instrucciones**

**Segmentación de cauce:**

## Conceptos básicos

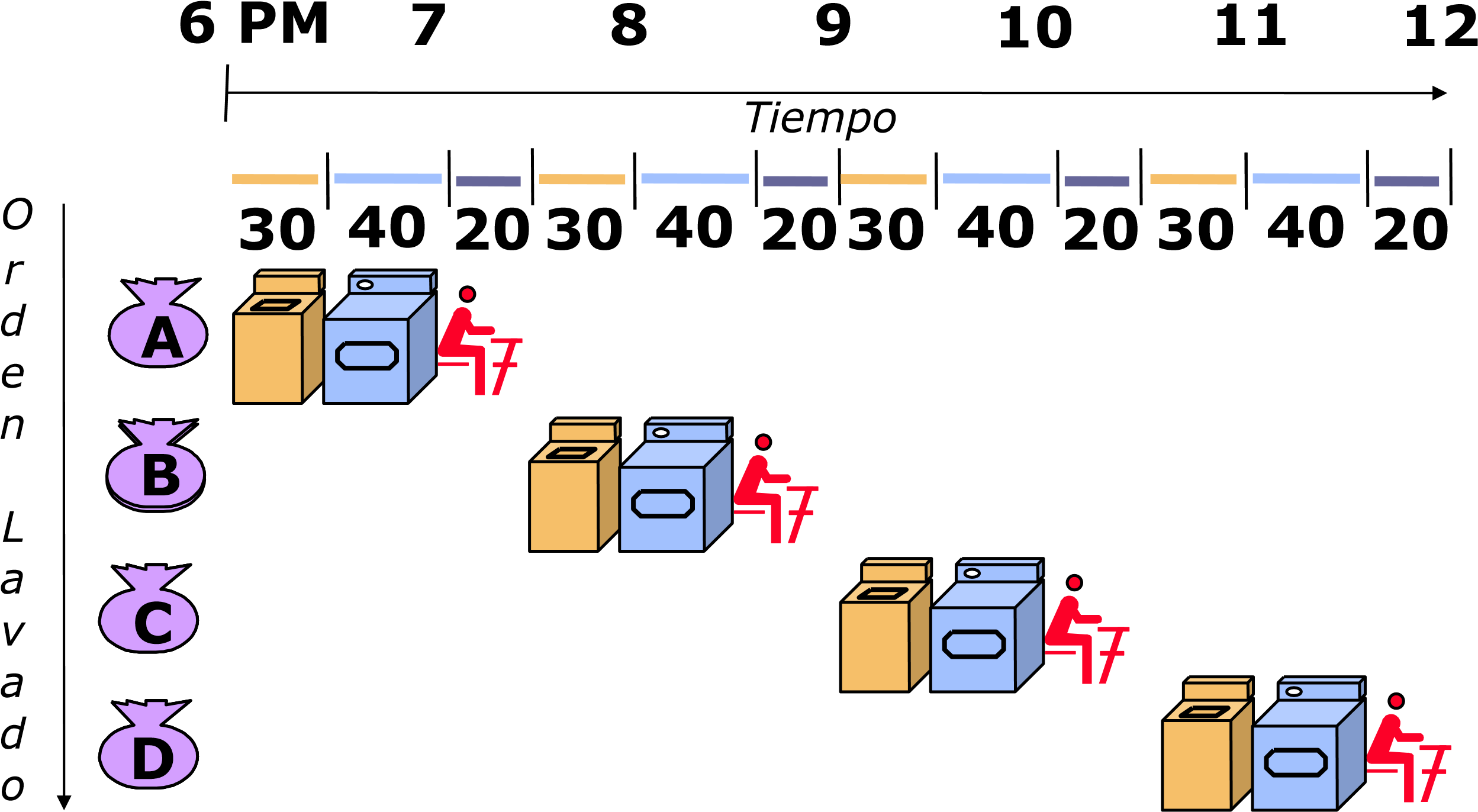
* La segmentación de cauce (pipelining) es una forma particularmente efectiva de organizar el hardware de la CPU para realizar más de una operación al mismo tiempo.
* Consiste en descomponer el proceso de ejecución de las instrucciones en fases o etapas que permitan una ejecución simultánea.
* Explota el paralelismo entre las instrucciones de un flujo secuencial.

# Ejemplo de estrategia (1)

* Similar a la línea de armado en una planta de manufactura.
* El producto pasa por varios estados en el proceso de producción.
* Por lo tanto, varios productos pueden ser manipulados simultáneamente (cada uno en estados distintos).
* Se puede comenzar el proceso nuevamente (entrada a la línea de producción) antes de que salga el producto final de la misma.

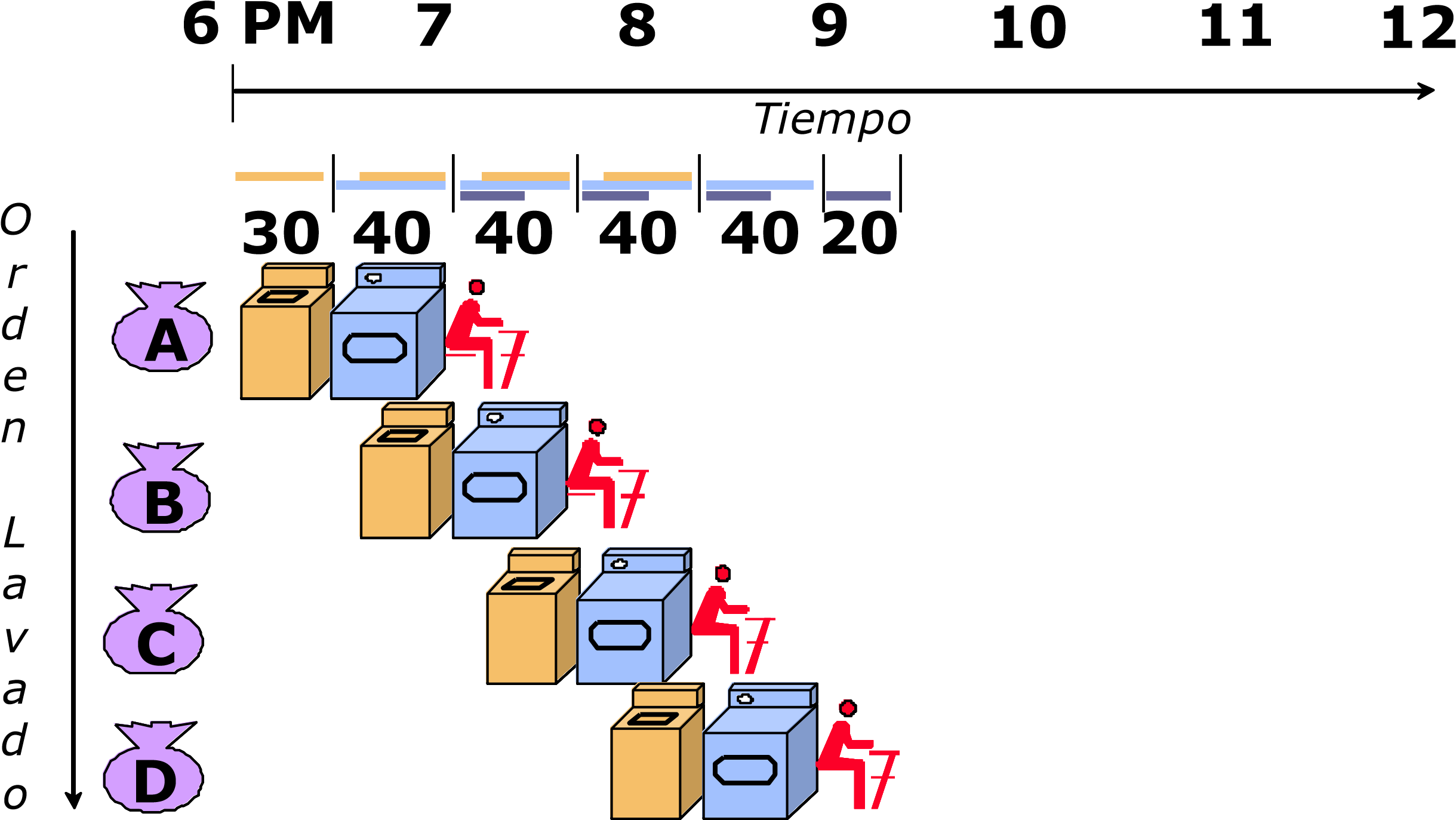
# Ej. de estrategia (2)

Lavandería secuencial: ¡mal negocio!



# Ej. de estrategia (3)

Lavandería segmentada: ¡buen negocio!

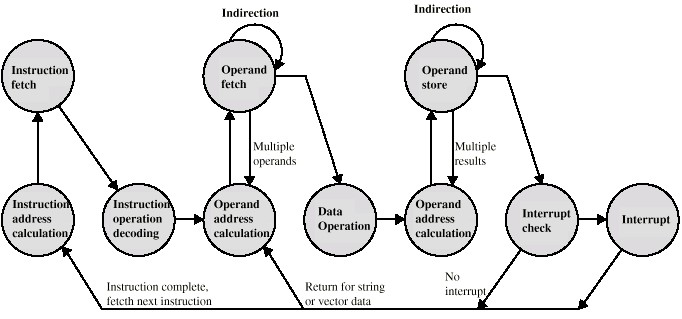


# Características

* La segmentación es una técnica de mejora de prestaciones a nivel de diseño hardware.
* La segmentación es invisible al programador.
* Necesidad de uniformizar las etapas.
* Al tiempo de la más lenta

* El diseño de procesadores segmentados tiene gran dependencia del repertorio de instrucciones.

# Diagrama de estados del ciclo de instrucción



**Captación de**

**instrucción**

**Captación**

**de**

**operando**

**Almacena**

**-**

**miento de**

**operando**

**Cálculo de**

**dir de prox**

**instrucción**

**Decodificac.**

**de operación**

**de la**

**instrucción**

**Cálculo de**

**la dirección**

**del**

**operando**

**Operación**

**con los**

**datos**

**Cálculo de**

**la dirección**

**del**

**operando**

**Interrupción?**

**Indirección**

**Indirección**

**Más de**

**un**

**resultado**

**Más de**

**un**

**operando**

**Instrucción finalizada,**

**captar la instrucción siguiente**

**Volver por otro dato**

**cadena o vector**

**No**

**interrupción**

**Interrupción**

# Tareas a realizar por ciclo

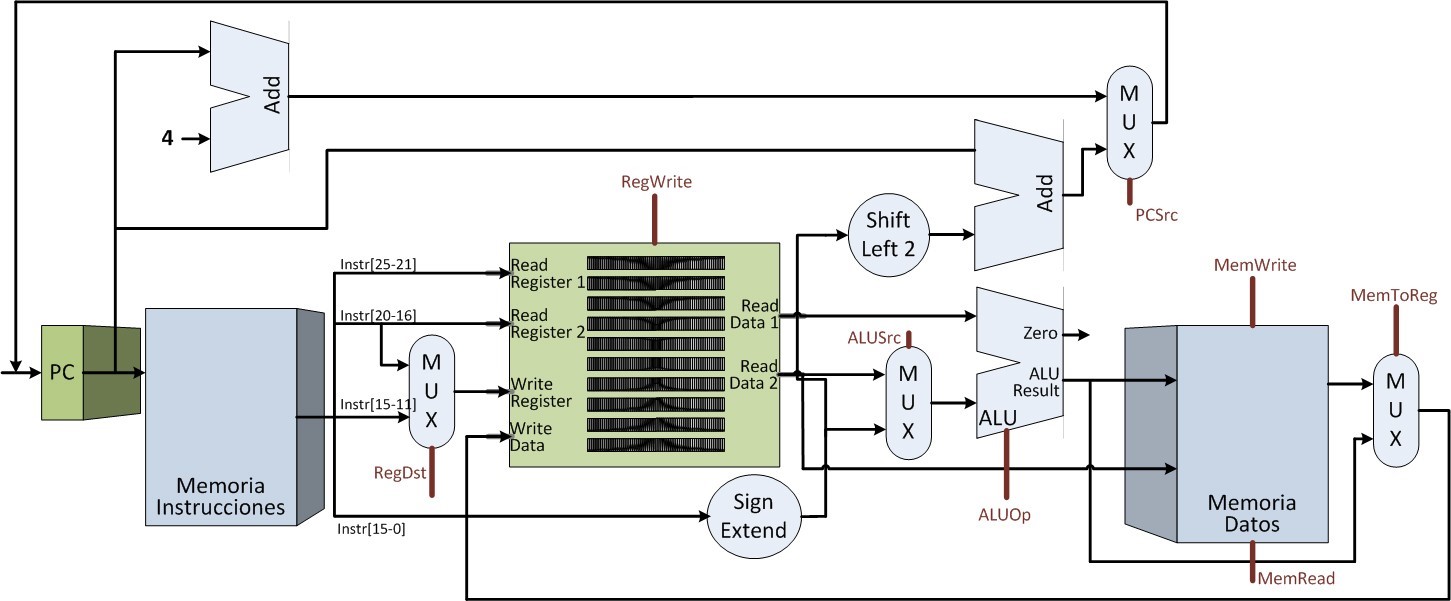
## • Búsqueda (F, Fetch)

* Se accede a memoria por la instrucción
* Se incrementa el PC

## • Decodificación (D, Decode)

* Se decodifica la instrucción, obteniendo operación a realizar en la ruta de datos
* Se accede al banco de registros por el/los operando/s (si es necesario)
* Se calcula el valor del operando inmediato con extensión de signo (si hace falta)
* **Ejecución (X, Execute)**
* Se ejecuta la operación en la ALU
* **Acceso a memoria (M, Memory Access)**
* Si se requiere un acceso a memoria, se accede
* **Almacenamiento (W, Writeback)**
* Si se requiere volcar un resultado a un registro, se accede al banco de registros

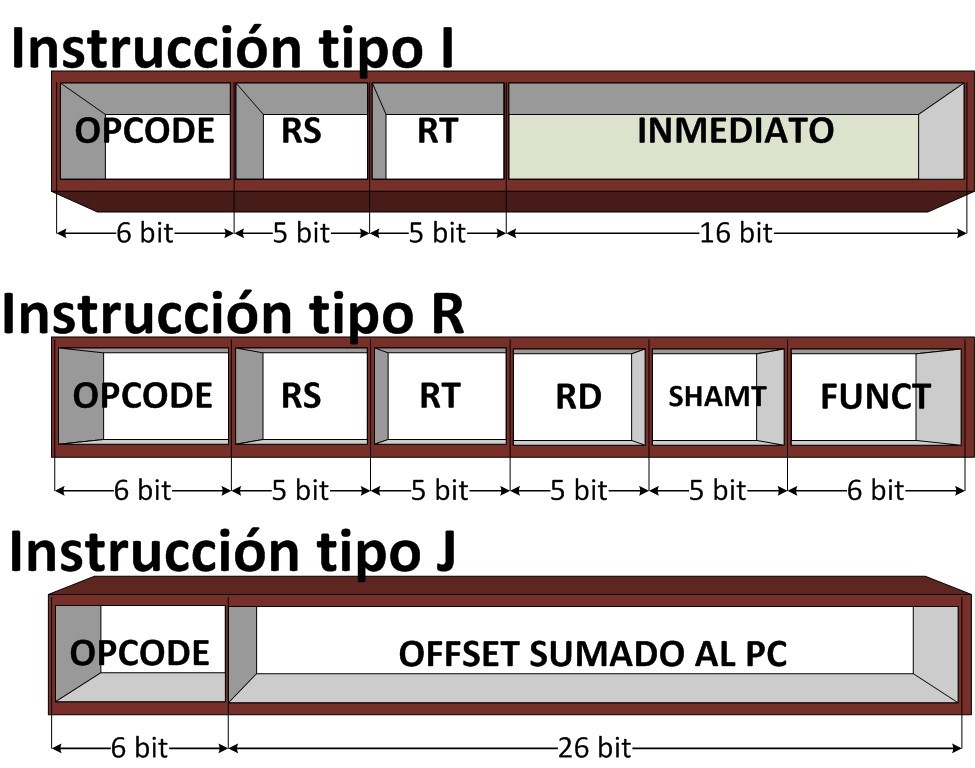
# Ruta de Datos en un ciclo



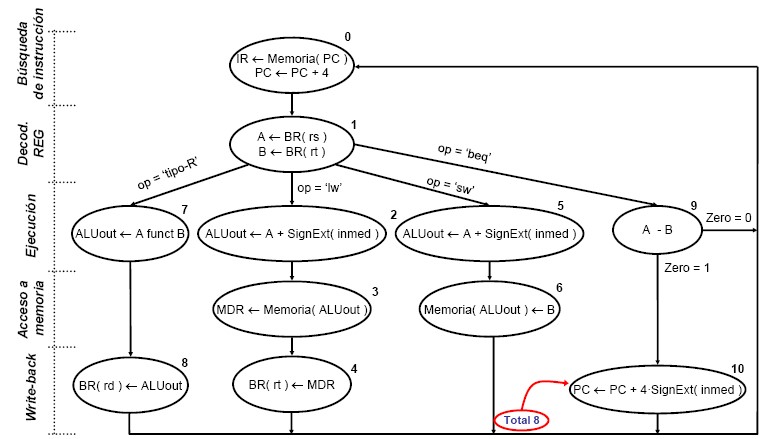
**Repertorio sencillo de instrucciones**

|  |  |  |
| --- | --- | --- |
| **Instrucción** | **Pseudocódigo** | **Descripción** |
| **LW** | LW RT, inmed(RS) | Carga registro RT desde memoria |
| **SW** | SW RT, inmed(RS) | Almacena en memoria desde registro RT |
| **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 |
| **BEQ** | BEQ RS, RT, destino | Salta a ‘destino’ si RS es igual a RT |

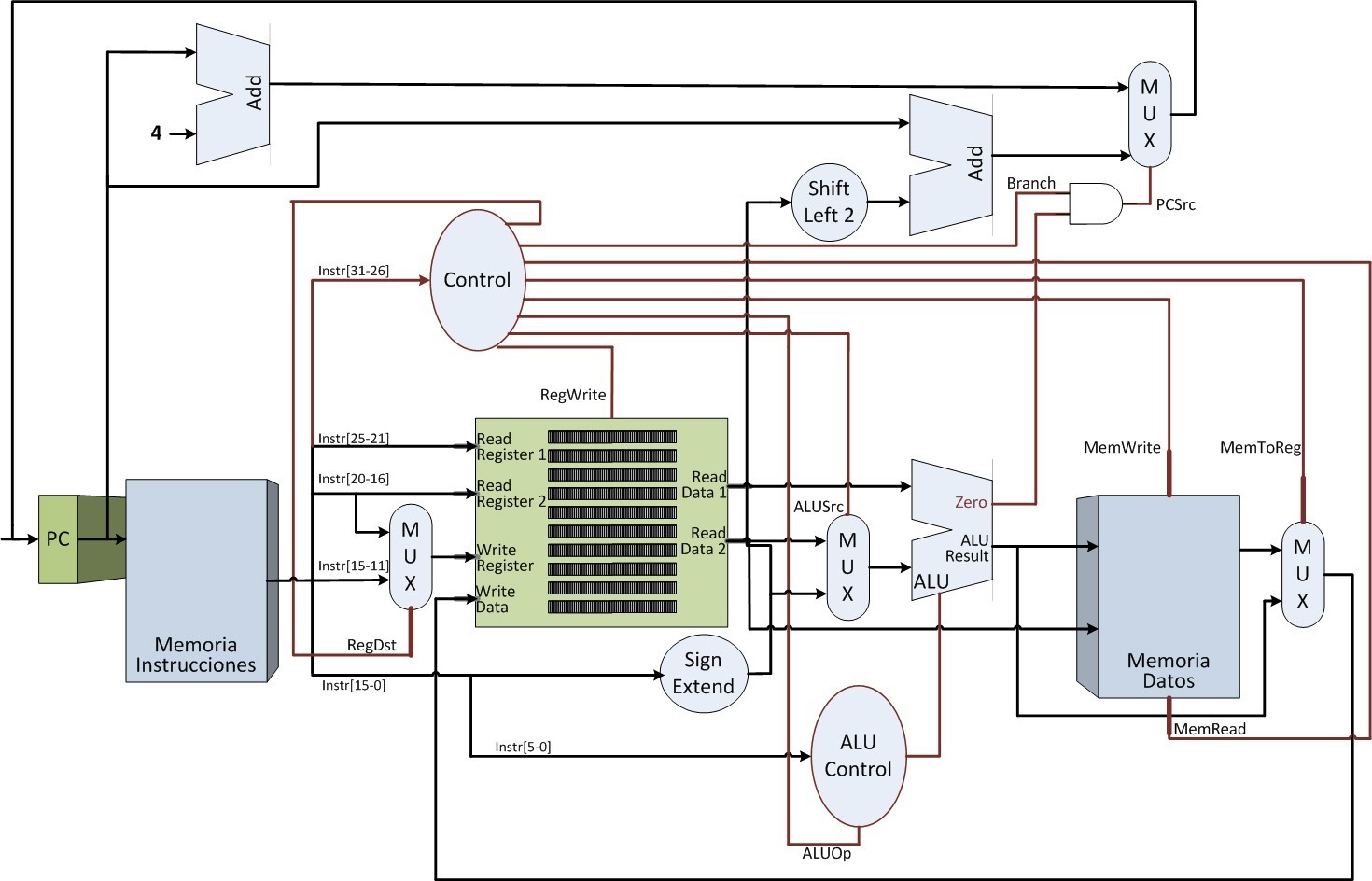
# Formato de instrucción



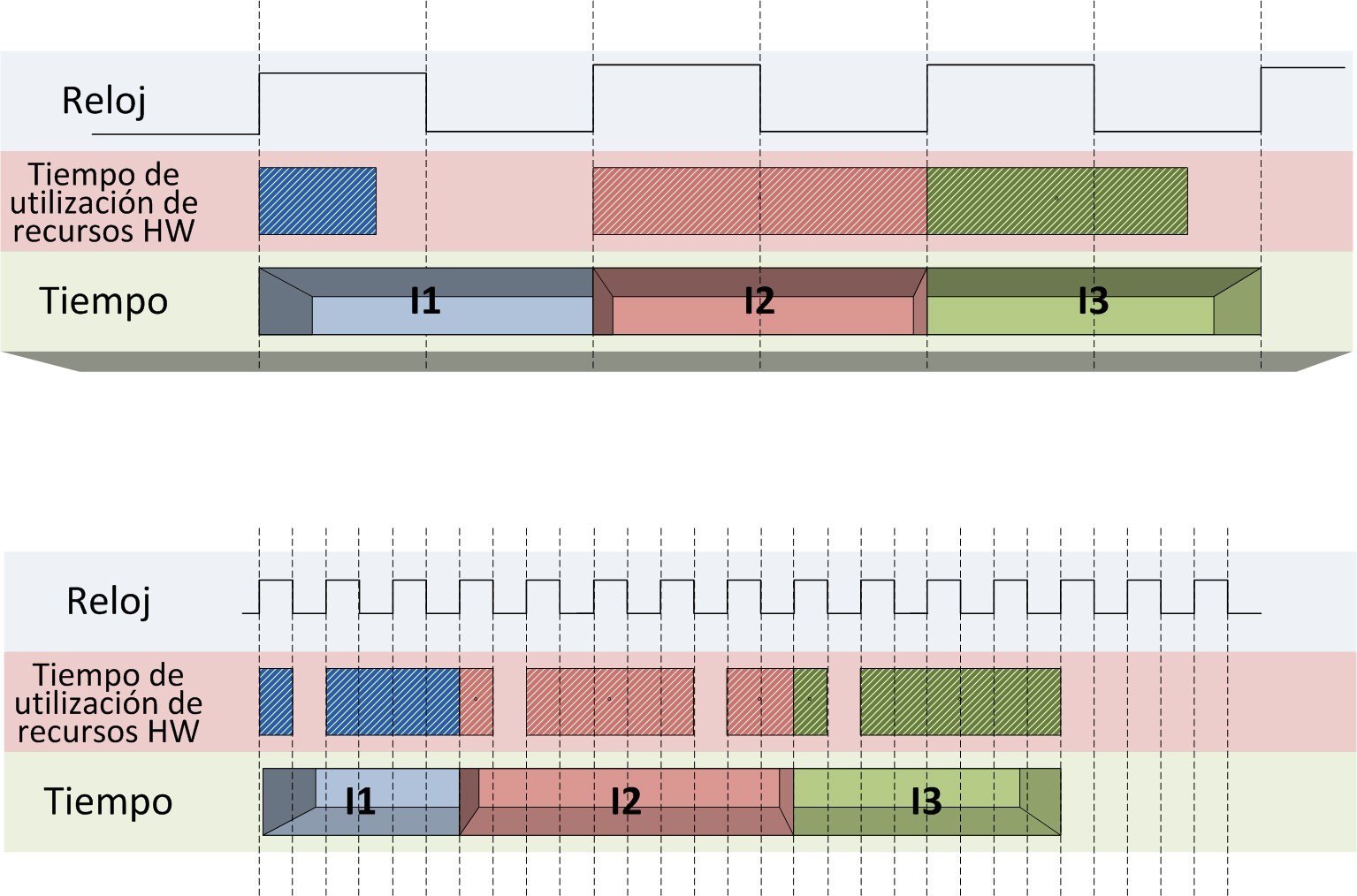
## Diagrama de estados del controlador



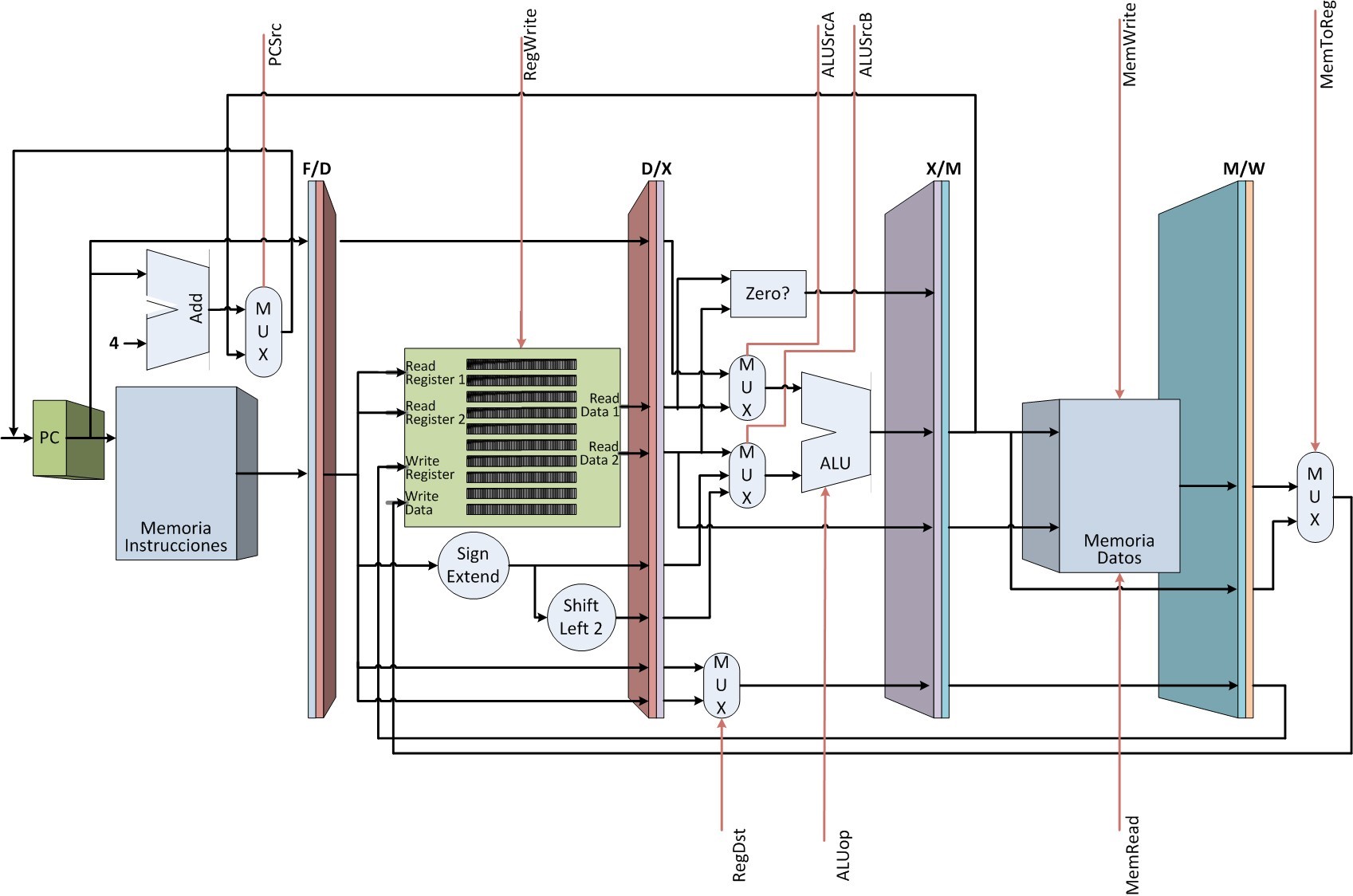
## Ruta de Datos y unidad de control



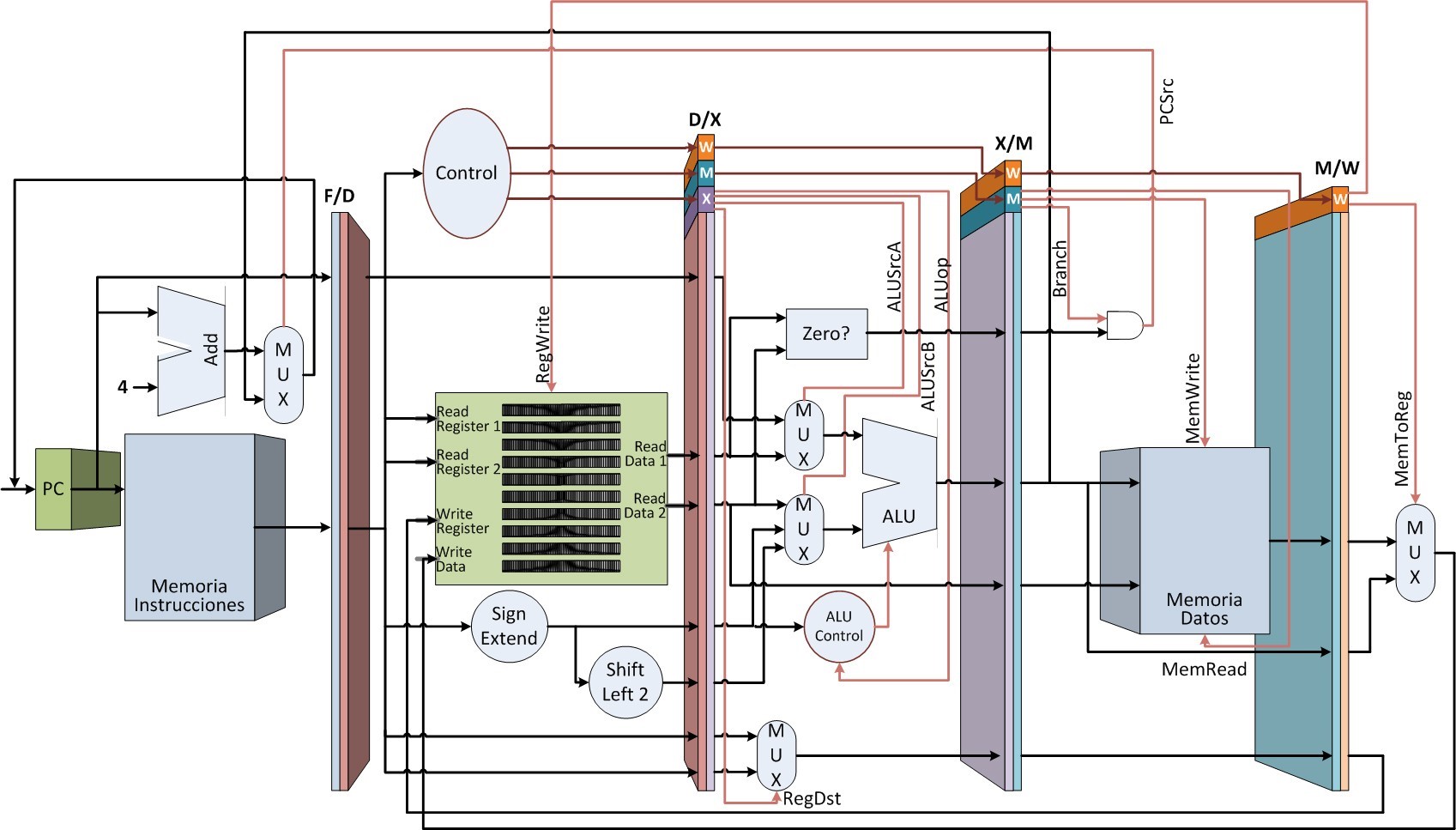
## Comparación monociclo-multiciclo



# Ruta de datos segmentados



## Ruta de datos y control segmentado



## Prestaciones del cauce segmentado

**Teórica**: El máximo rendimiento es completar una instrucción con cada ciclo de reloj.

Si K es el número de etapas del cauce 

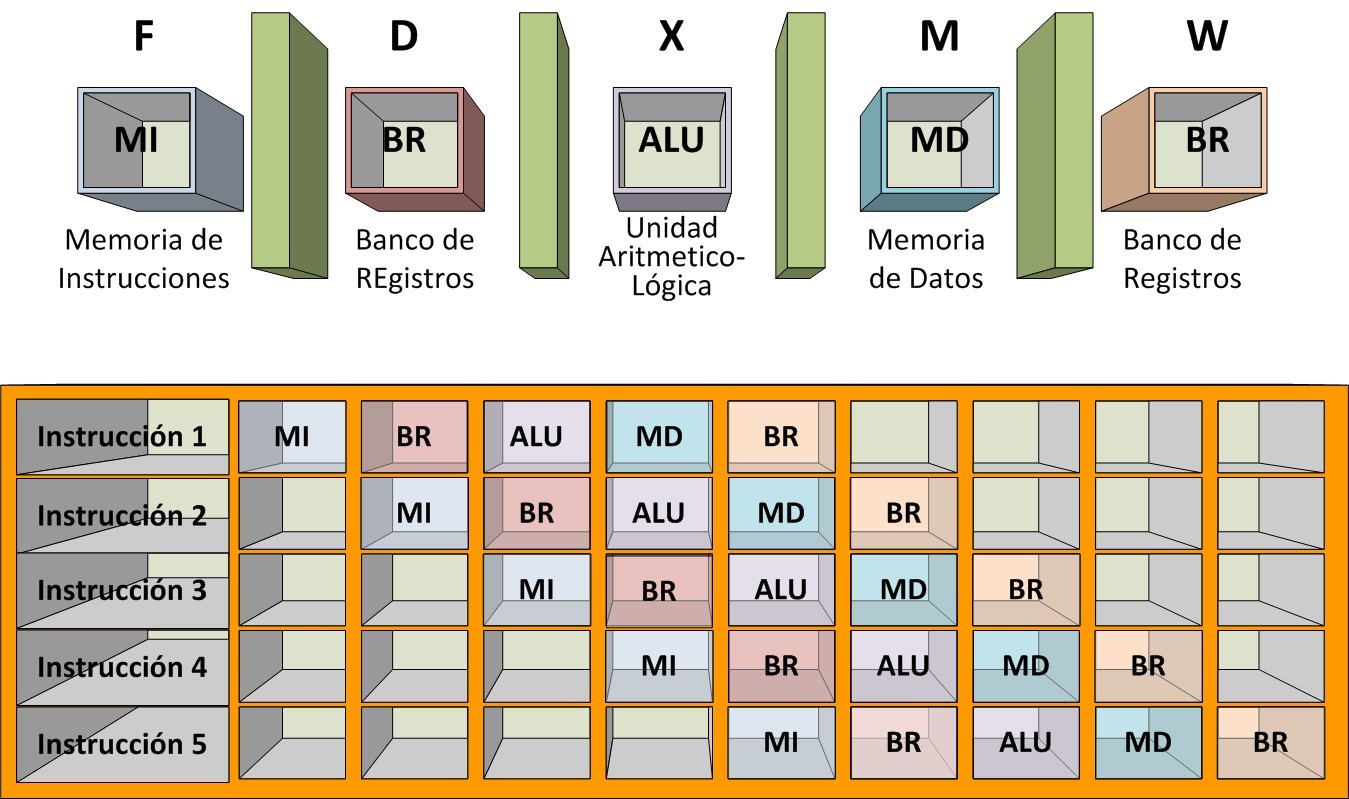
Vel. procesador segmentado = Vel. secuencial x K

El incremento potencial de la segmentación del cauce es proporcional al número de etapas del cauce.

**Incrementa la productividad (throughput), pero no reduce el tiempo de ejecución de la**

**instrucción**

# Ejemplo de segmentación



# Análisis de la segmentación (1)

Suposiciones:

•Todas las tareas duran el mismo tiempo.

•Las instrucciones siempre pasan por todas las etapas.

•Todos las etapas pueden ser manejadas en paralelo.

# Análisis de la segmentación (2)

Problemas:

•No todas las instrucciones necesitan todas las etapas.

•SW RT, inmed(RS) ; no utiliza W

• En MSX88: un MOV AX, mem ; no requiere X

•No todas las etapas pueden ser manejadas en paralelo.

•F y M acceden a memoria

•No se tienen en cuenta los saltos de control.

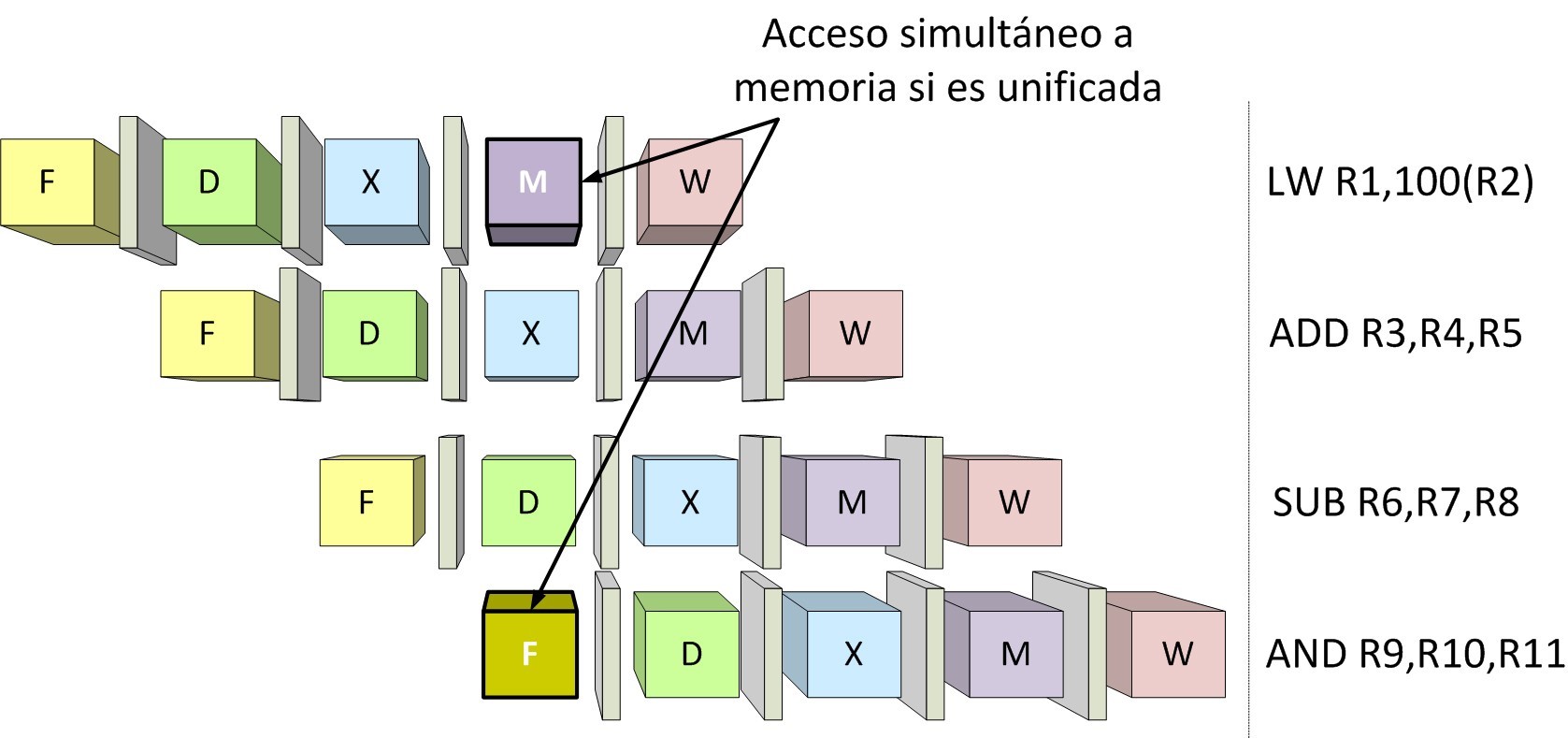
# Atascos de un cauce (stall)

Situaciones que impiden a la siguiente instrucción que se ejecute en el ciclo que le corresponde.

* Estructurales
* Provocados por conflictos por los recursos
* Por dependencia de datos
* Ocurren cuando dos instrucciones se comunican por medio de un dato (ej.: una lo produce y la otra lo usa)
* Por dependencia de control
* Ocurren cuando la ejecución de una instrucción depende de cómo se ejecute otra (ej.: un salto y los 2 posibles caminos)

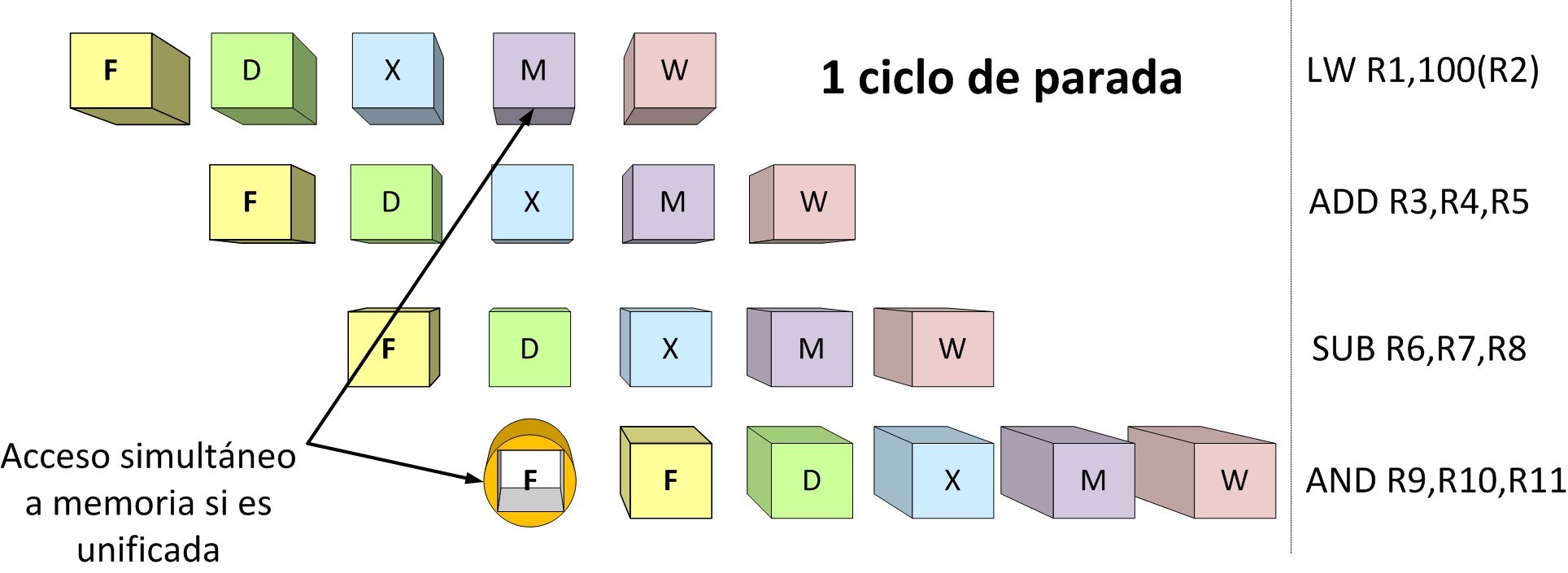
# Riesgos estructurales

Dos o mas instrucciones necesitan utilizar el mismo recurso hardware en el mismo ciclo.



# Riesgos estructurales (2)

Resolución ante el riesgo:



## Riesgos por dependencias de datos

• Condición en la que los operandos fuente o destino de una instrucción no están disponibles en el momento en que se necesitan en una etapa determinada del cauce.

# Tipos de dependencias de datos

* Lectura después de Escritura (RAW, dependencia verdadera)
* una instrucción genera un dato que lee otra posterior

* Escritura después de Escritura (WAW, dependencia en salida)
* una instrucción escribe un dato después que otra posterior
* sólo se da si se deja que las instrucciones se adelanten unas a otras

* Escritura después de Lectura (WAR, antidependencia)
* una instrucción modifica un valor antes de que otra anterior que lo tiene que leer, lo lea
* no se puede dar en nuestro cauce simple

# Tipos de dependencias ...(2)

**Ri  ...**

**RAW** Dependencia real

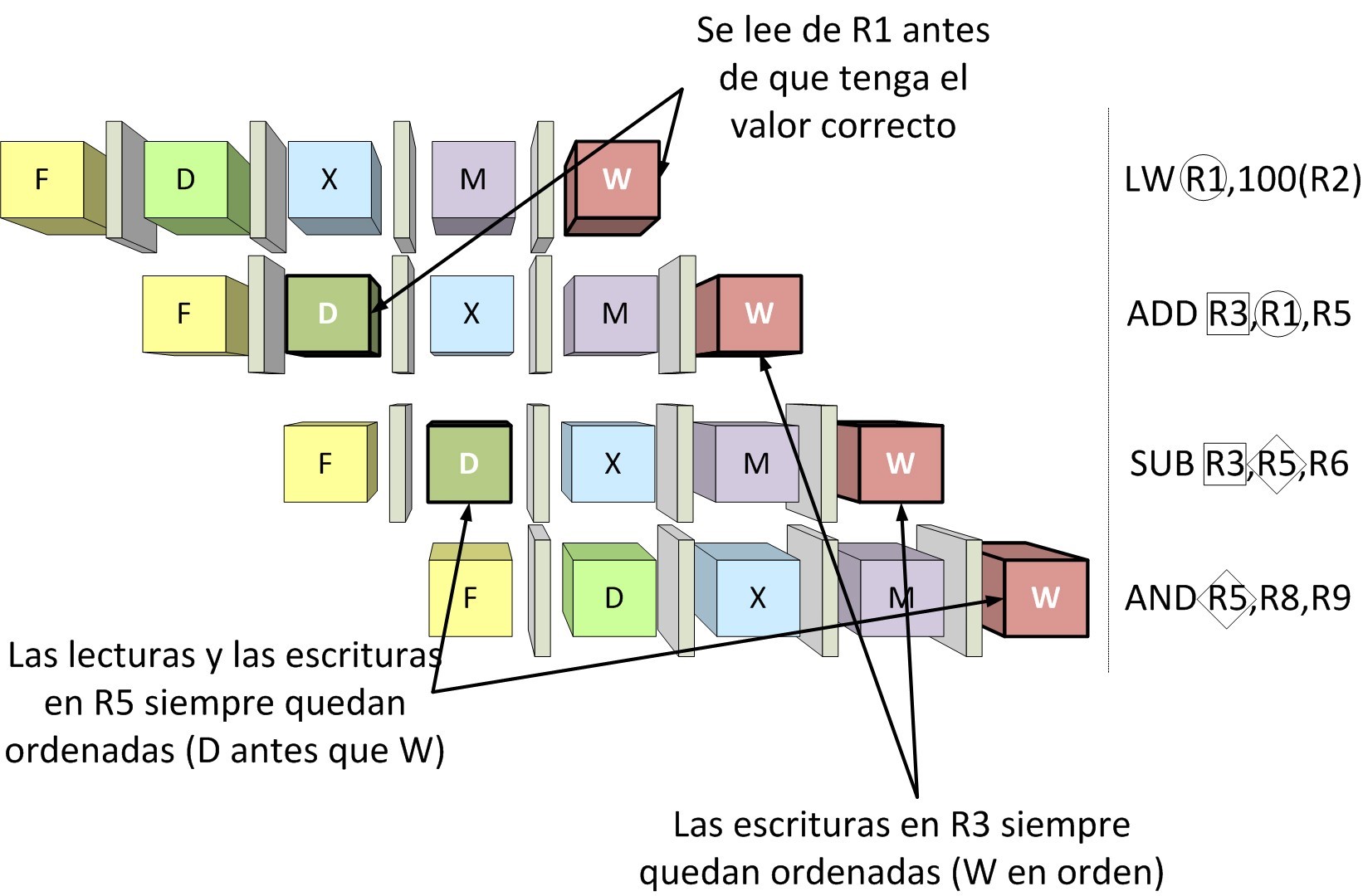
Dependencia de salida **WAW** **Ri op Rj**

**...**

**WAR** Antidependencia

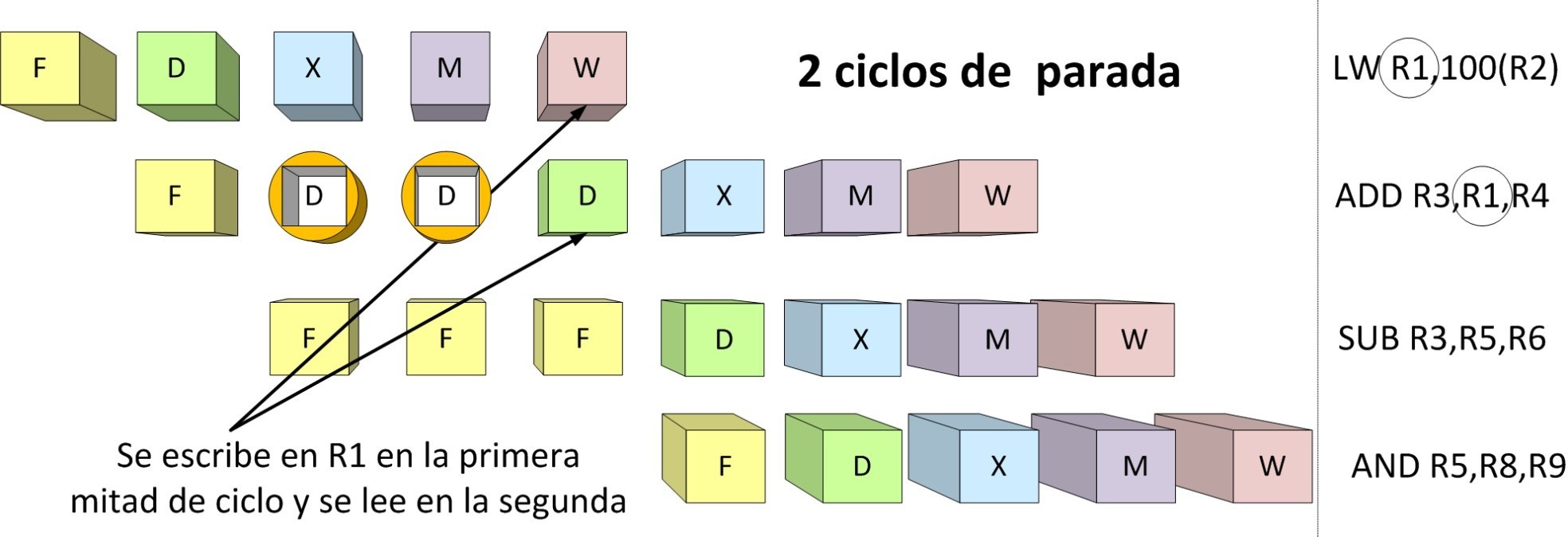
**Ri  ...**

**Riesgos por dep… datos (2)**



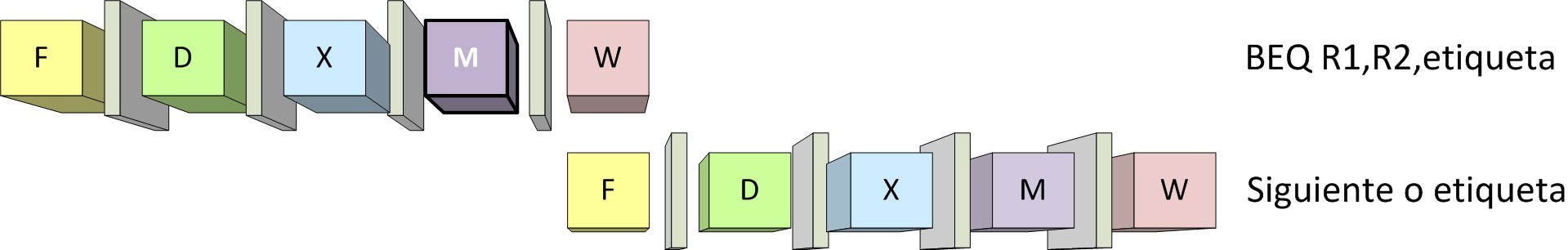
**Riesgos por dep… datos (3)**

Resolución ante el riesgo:



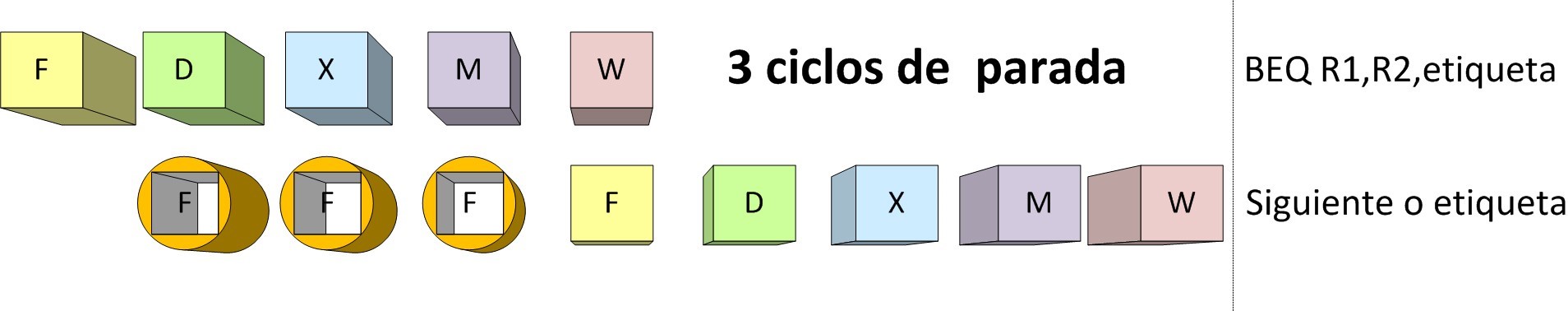
**Riesgos de control (o de instrucciones)**

Una instrucción que modifica el valor del PC no lo ha hecho cuando se tiene que comenzar la siguiente.



# Riesgos de control (2)

Resolución ante el riesgo:



# Lectura básica

* Organización y Arquitectura de Computadores, W. Stallings, Capítulo 11, 5ta ed.
* Diseño y evaluación de arquitecturas de computadoras, M. Beltrán y A. Guzmán, Capítulo 1, 1er ed.