[Resumen clase 4](https://drive.google.com/file/d/18LMy90hBdiboFCVbqLQfY4Xvj5Jyc7EH/view?usp=sharing)

Segmentación de instrucciones

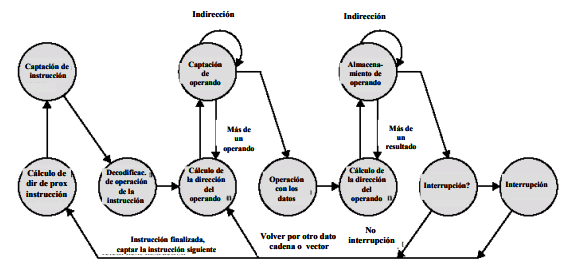
# Segmentación de cauce (Pipelining)

* 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 permiten una ejecución simultánea
* Explota el paralelismo entre las instrucciones de flujo secuencial

Similar a una línea de armado en una planta de manufactura, donde el producto pasa por varios estados en el proceso, por lo que varios productos pueden ser manipulados simultáneamente (cada uno en estados distintos) pudiendo comenzar el proceso nuevamente antes de que salga el producto final de la misma.

## Características

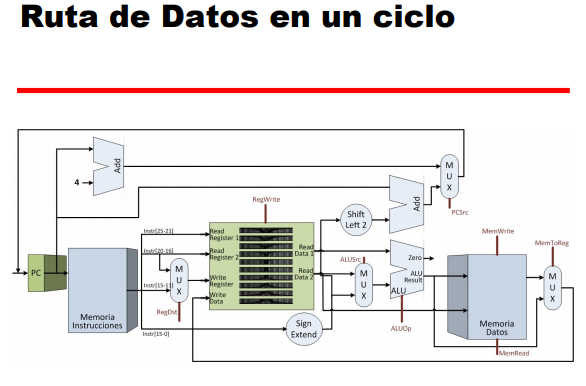
* La segmentación es una técnica de mejora de prestaciones a nivel de diseño hardware
* La segmentación es invisible al programador
* Necesidad de uniformizar las etapas (al tiempo de la mas lenta)
* El diseño de procesadores segmentados tiene gran dependencia del repertorio de instrucciones

Diagrama de estados del ciclo de instrucción

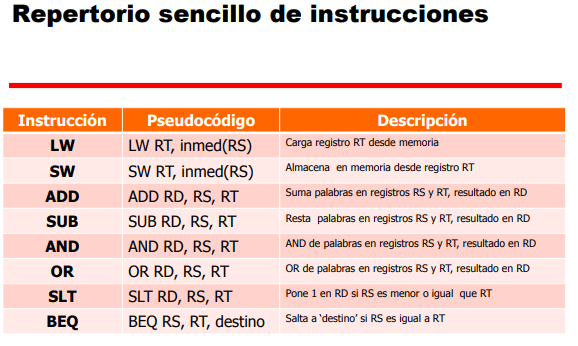
## Tareas a realizar por ciclo

* 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

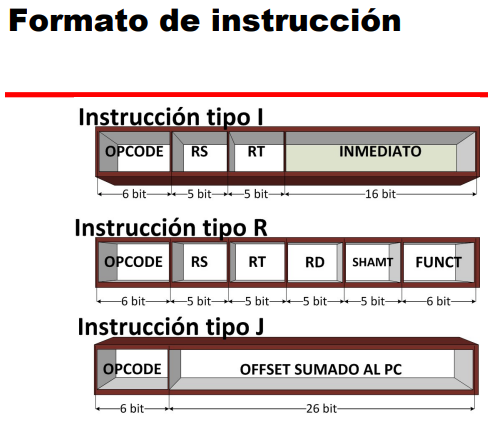
## Ruta de datos de un ciclo



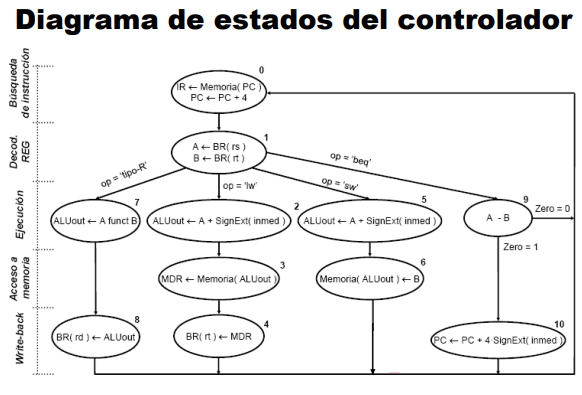
## Repertorio sencillo de instrucciones

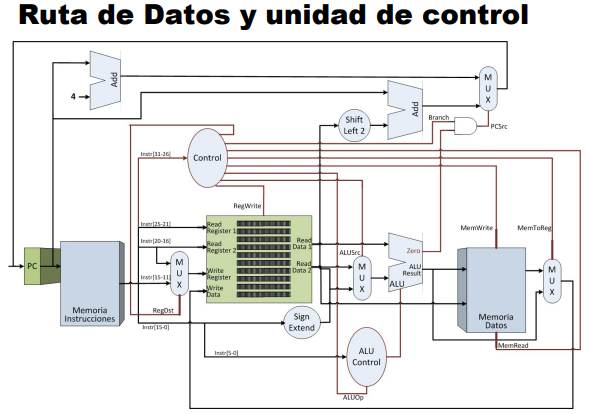


## Formato de instrucción

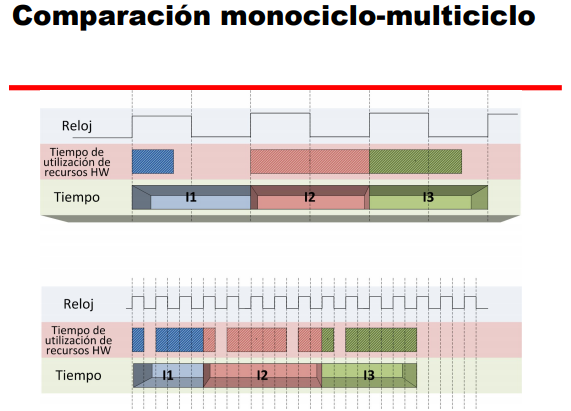


## Diagrama de estados del controlador

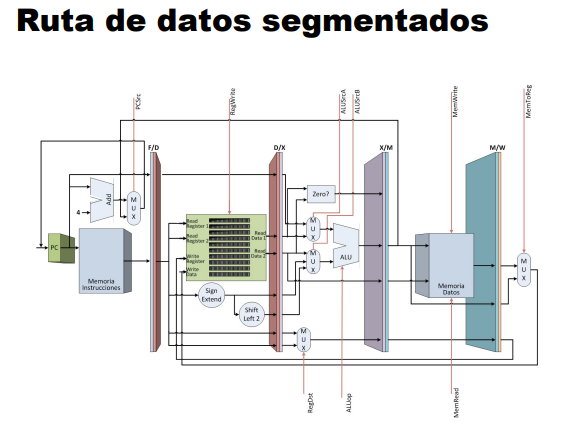




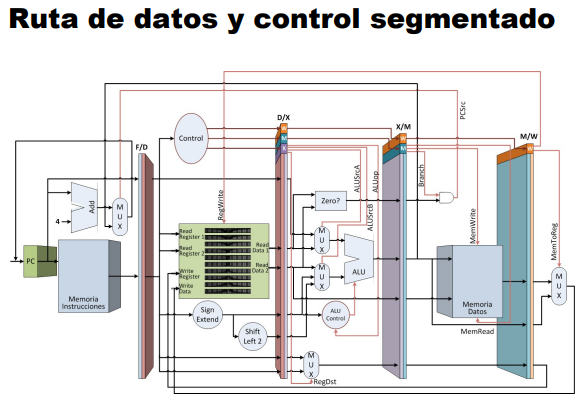
## Comparación monociclo-multiciclo



## Ruta de datos segmentados



## Ruta de datos y control segmentado



## Prestaciones del cauce segmentado

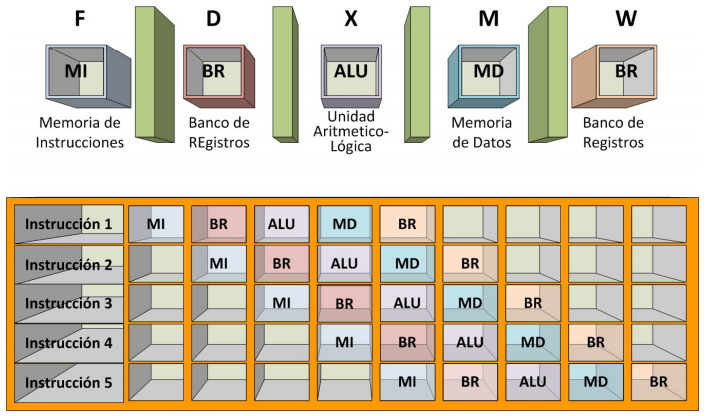
* Teórica: El máximo rendimiento es completar una instrucción con cada ciclo de reloj

Si K es el número de etapas del cauce→ Velocidad procesador segmentado = Velocidad 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

Suposiciones:

* Todas las tareas duran el mismo tiempo
* Las instrucciones siempre pasan por las mismas etapas
* Todas las etapas pueden ser manejadas en paralelo

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 tiene en cuenta los saltos de control

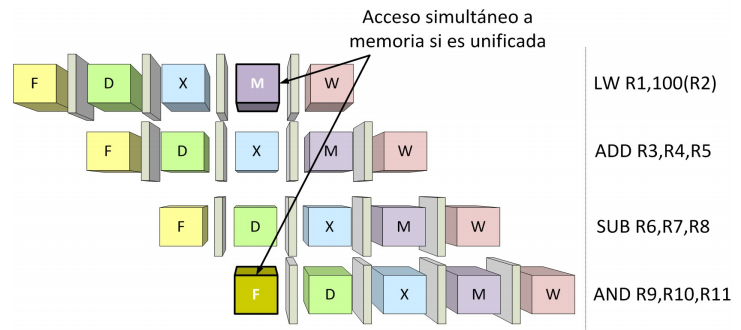
## Atasco de un cauce (Stall)

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

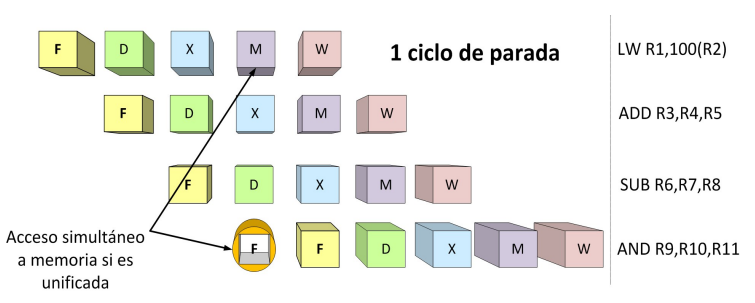
* Estructurales: provocados por conflictos de recursos
* Por dependencia de datos: ocurren cuando dos instrucciones se comunican por medio de un dato (ej: una lo produce y otra lo usa)
* Por dependencia de control: ocurren cuando la ejecución de una instrucción depende de cómo se ejecute la otra (ej: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

### Riesgos estructurales

Dos o más instrucciones necesitan usar el mismo recurso hardware en el mismo ciclo



Resolución ante el riesgo:

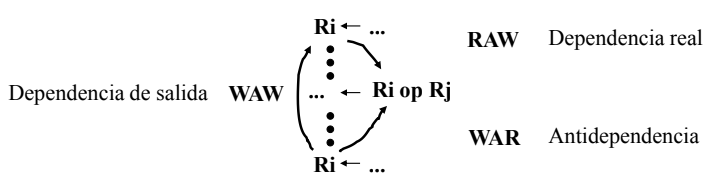


### Riesgos por dependencia de datos

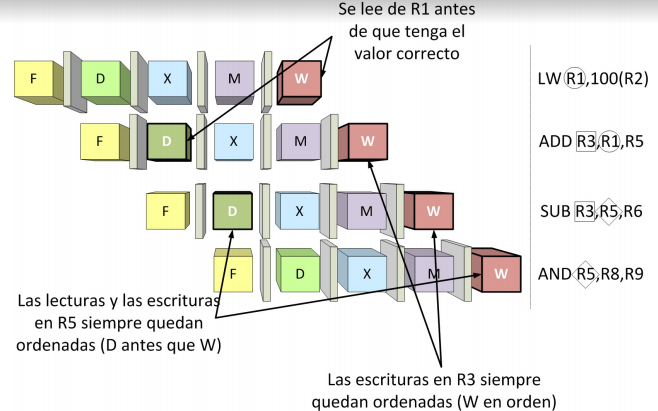
Condición en la que los operando fuente o destino de una instrucción no están disponibles en el momento en que se necesitan en una etapa determinada del cauce.

Tipos:

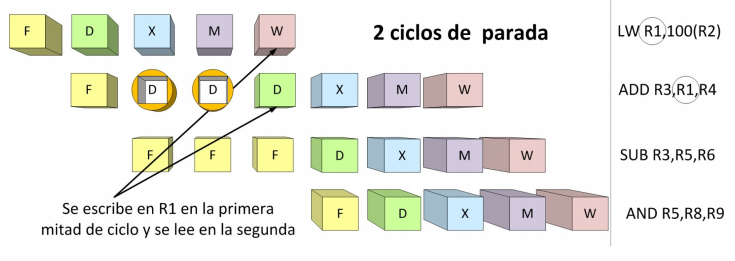
* Lectura despues de escritura (RAW, dependencia verdadera): una instrucción genera un dato que lee otra posterior
* Escritura después de escritura (WAW, dependencia en salida): una instrucción escribe un dato después que otra posterior. Solo se da si las instrucciones se adelantan unas a otras
* Escritura 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.



Riesgos por dependencia de datos:

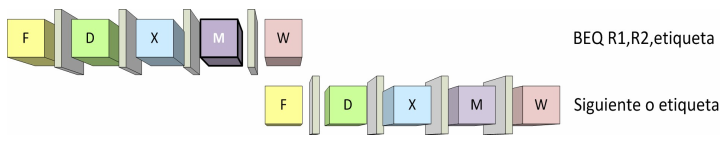


Resolución ante el riesgo:



### Riesgos de control (o de instrucciones)

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



Resolución ante el riesgo:

