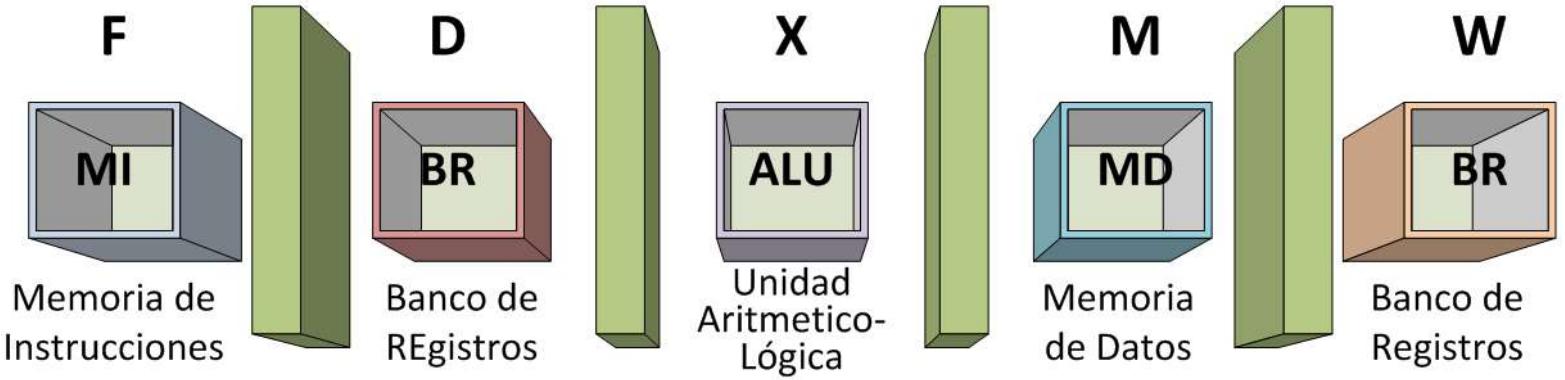
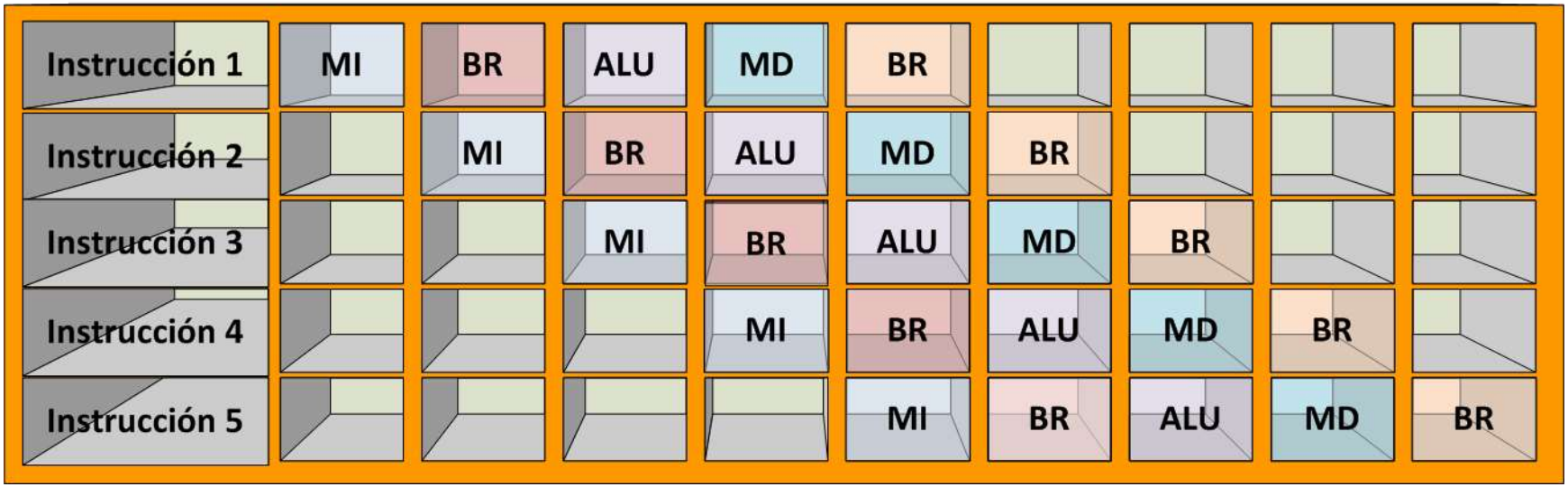
Arquitectura de Computadoras

Clase 6

Posibles soluciones a atascos

Ejemplo de segmentación





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
  - Dos instrucciones se comunican por medio de un dato
- Por dependencia de control
  - La ejecución de una instrucción depende de cómo se ejecute otra

Si resolvemos con paradas del cauce, disminuye el rendimiento teórico y CPI>1.

Soluciones a riesgos estructurales

Simple: Replicar, segmentar ó realizar turnos para el acceso a las unidades funcionales en conflicto.

- Duplicación de recursos hardware
  - Sumadores o restadores además de la ALU
- Separación en memorias de instrucciones y datos
- Turnar el acceso al banco de registros
  - Escrituras en la 1º mitad de los ciclos de reloj
  - Lecturas en la 2º mitad de los ciclos de reloj

Soluciones a riesgos de datos

Para riesgos RAW: se debe determinar cómo y cuando aparecen esos riesgos

Será necesario

- Unidad de detección de riesgos y/o compilador mas complejo

Dos soluciones:

- **Hardware**
  - Adelantamiento de operandos (forwarding)
- **Software**
  - Instrucciones NOP o reordenación de código

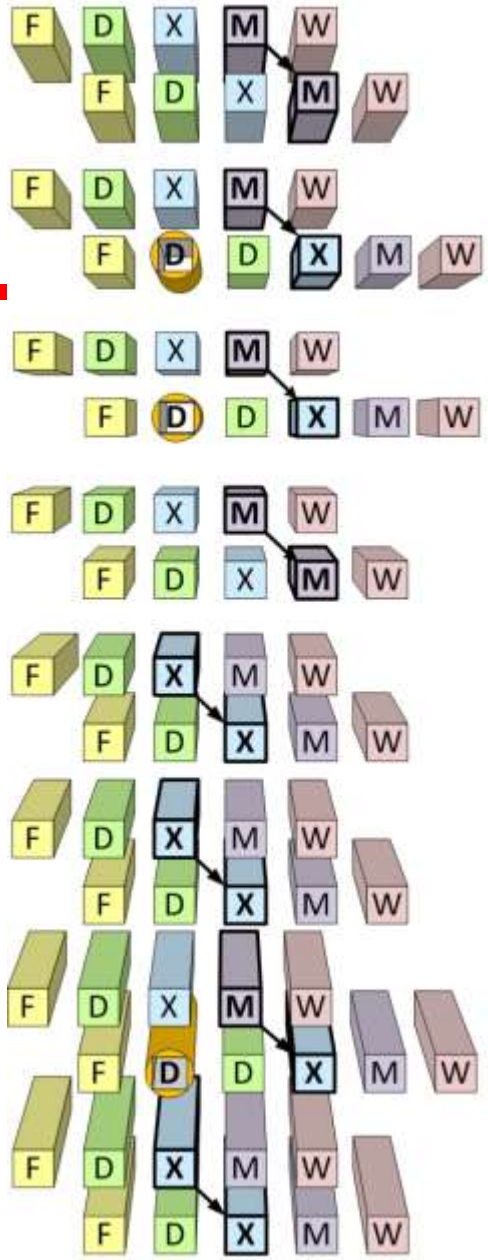
Soluciones a riesgos de datos (2)

Técnica hardware: conocida como

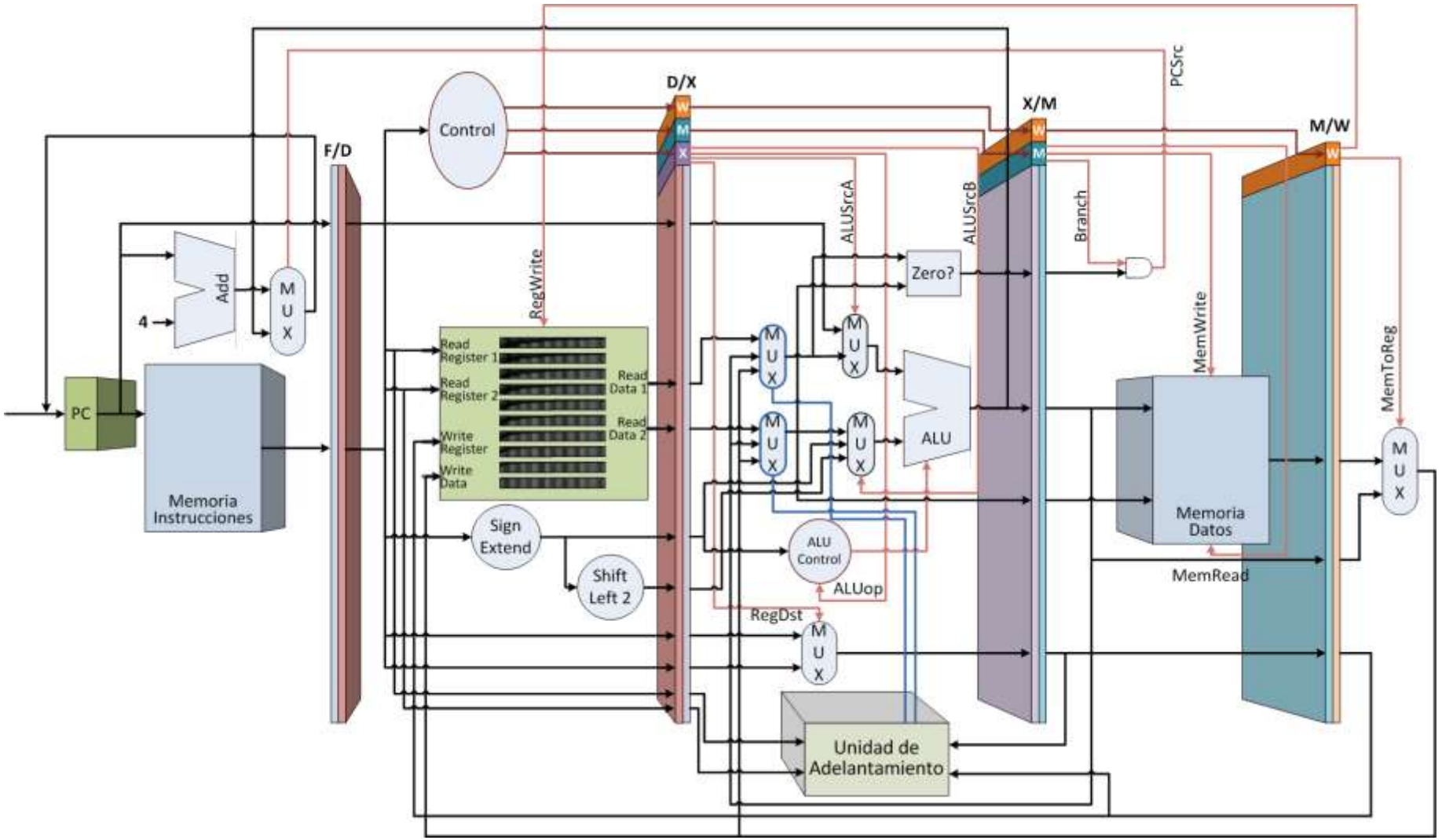
Adelantamiento, Forwarding o Cortocircuito

- Consiste en pasar directamente el resultado obtenido con una instrucción a las instrucciones que lo necesitan como operando.
  - Si el dato necesario está disponible a la salida de la ALU (X_i) se lleva a la entrada de la etapa correspondiente (X_{i+1}) sin esperar a la escritura (M_i o W_i).
- Fácil de implementar si se identifican todos los adelantamientos y se comunican a los registros de segmentación correspondientes.

Riesgos RAW


F D X M W
F D X M W
LW R1, 100(R2)
SW R1, 0(R10)
F D X M W
F D D X M W
LW R1, 100(R2)
ADD R3, R1, R8
F D X M W
F D D X M W
LW R1,100(R2)
BEQ R1, R3,etiqueta
F D X M W
F D X M W
ADD R1, R2, R3
SW R1, 0(R10)
F D X M W
F D X M W
ADD R1, R2, R3
SUB R8, R1, R6
F D X M W
F D X M W
ADD R1, R2, R3
BEQ R1, R7, etiqueta
F D X M W
F D D X M W
LW R1,0(R7)
LW R10, 100(R1)
F D X M W
F D X M W
ADD R1, R2, R3
LW R10, 100(R1)

Ruta de datos con adelantamiento



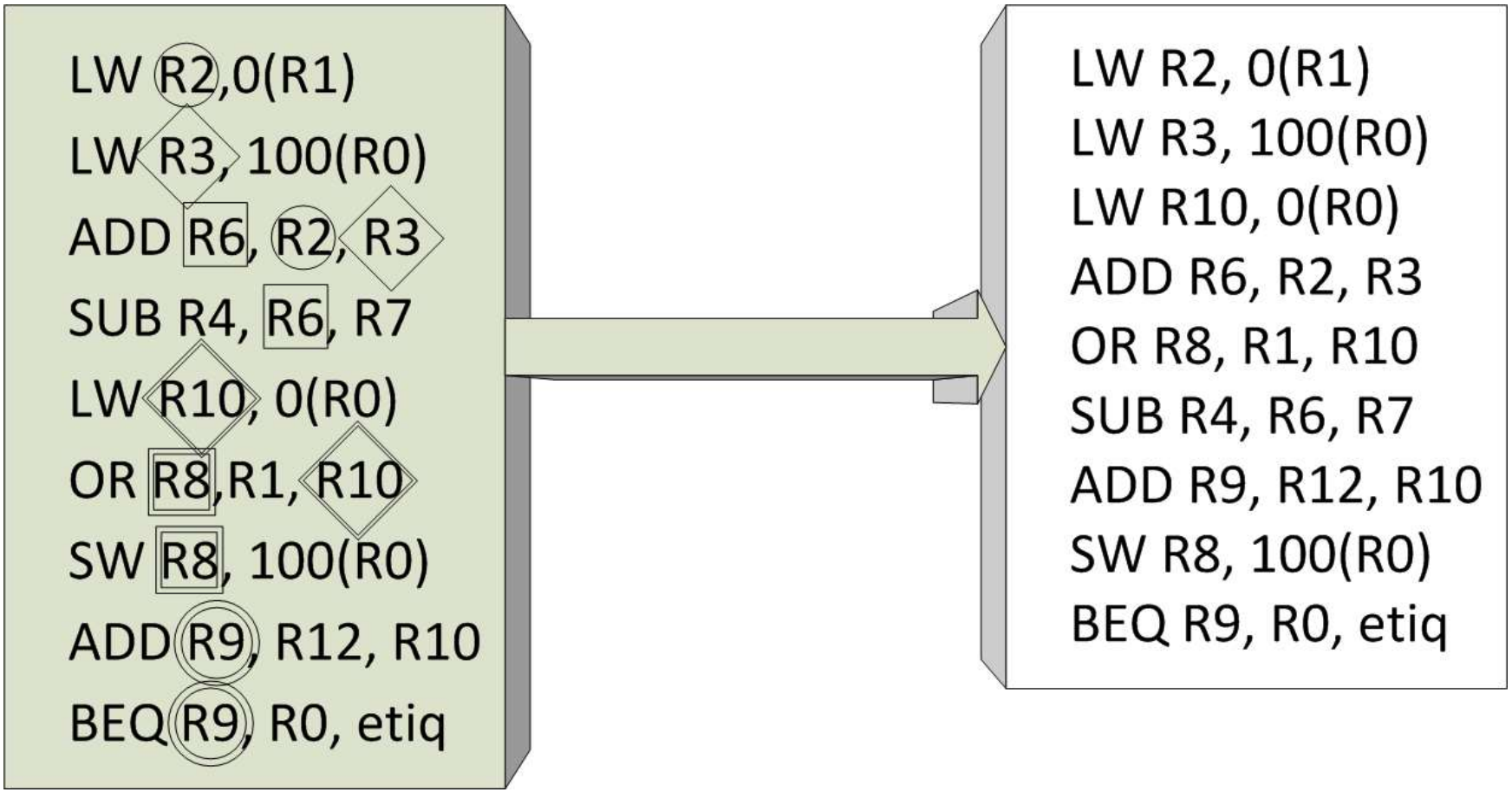
Soluciones a riesgos de datos (3)

Técnica software: evita los riesgos reordenando las instrucciones del código sin afectar los resultados

Realizada por el compilador

- Introducción de instrucciones NOP
  - Se genera Retardo
- Reordenación de instrucciones
  - Máxima separación de instrucciones con dependencia RAW
  - Cuidado con ejecución "fuera de orden"

Reordenación por compilador



Soluciones a riesgos de control

Existe una Penalización por salto

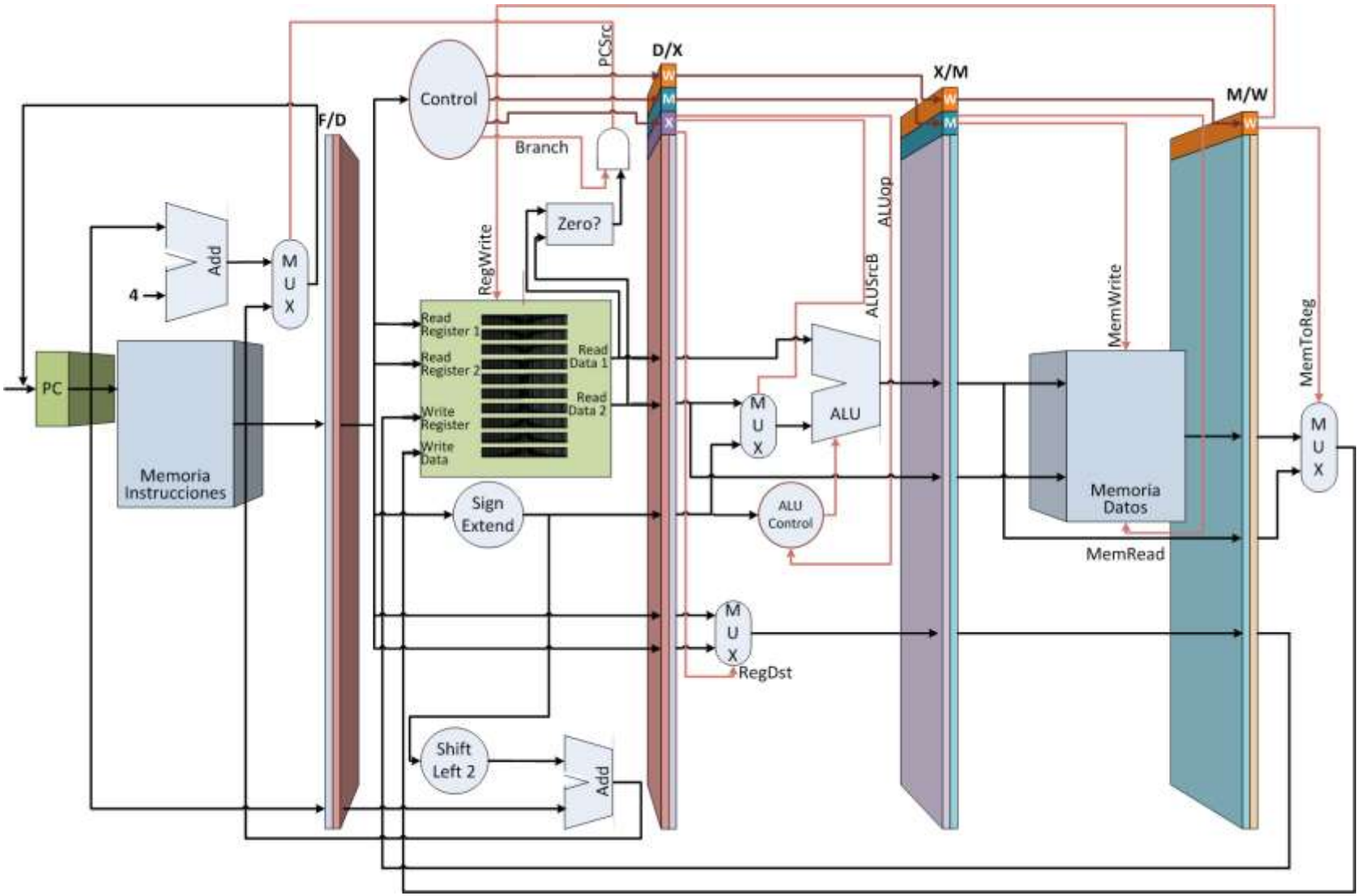
- Instrucciones de salto
  - Incondicional
    - La dirección de destino se debe determinar lo más pronto posible, dentro del cauce, para reducir la penalización.
  - Condicional
    - Introduce riesgo adicional por la dependencia entre la condición de salto y el resultado de una instrucción previa.

Soluciones a riesgos de control (2)

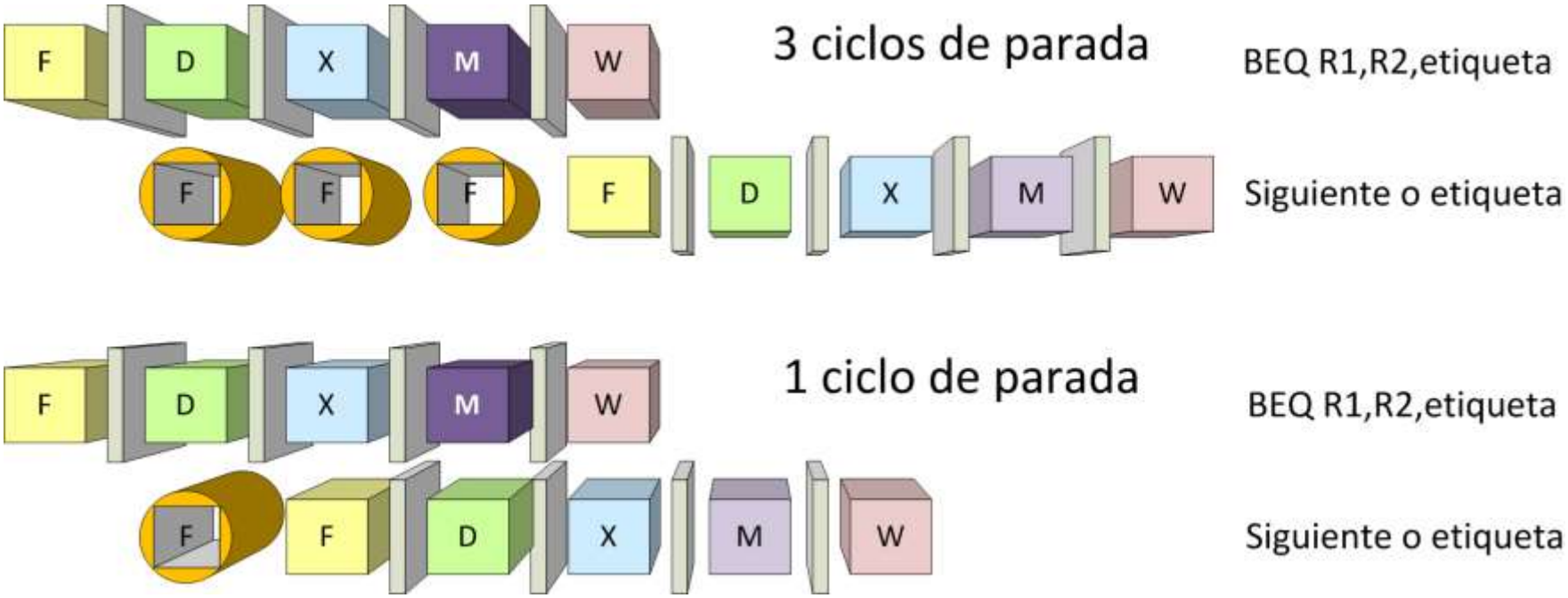
Modificación sencilla de la ruta de datos para reducir la cantidad de paradas a un solo ciclo.

- Adelantar la resolución de los saltos a la etapa D
  - En ella se decodifica y se sabe que es un salto
  - Se puede evaluar la condición de salto (con restador)
  - Se puede calcular la dirección de salto (con sumador)

Ruta de datos mejorada



Reducción de paradas por mejora



Soluciones a riesgos de control (3)

Para tratamiento de saltos hay

Técnica Hardware

- Predicción de saltos para evitar la parada

Técnica Software

- Salto retardado o de relleno de ranura de retardo
  - El compilador introduce instrucciones que se ejecutarán en cualquier caso después de la instrucción de salto

Predicción de saltos (1)

- Técnicas estáticas
  - Predecir que nunca se salta:
    - Asume que el salto no se producirá.
    - Siempre capta la siguiente instrucción.
  - Predecir que siempre se salta:
    - Asume que el salto se producirá.
    - Siempre capta la instrucción destino del salto.

Ejemplo con Predicción nunca saltar

Predicción de saltos (2)

- Técnicas dinámicas
  - Conmutador saltar/no saltar:
    - Basado en la historia de las instrucciones.
    - Eficaz para los bucles.
  - Tabla de historia de saltos (branch-target buffer)
    - Pequeña cache asociada a la etapa de búsqueda (F)
    - Tres campos:
      - Dirección de una instrucción de bifurcación
      - Información de la instrucción destino
        - Dirección del destino ó Instrucción destino
      - N bits de estado (historia de uso)

Diagrama de flujo de predicción saltar/ no saltar

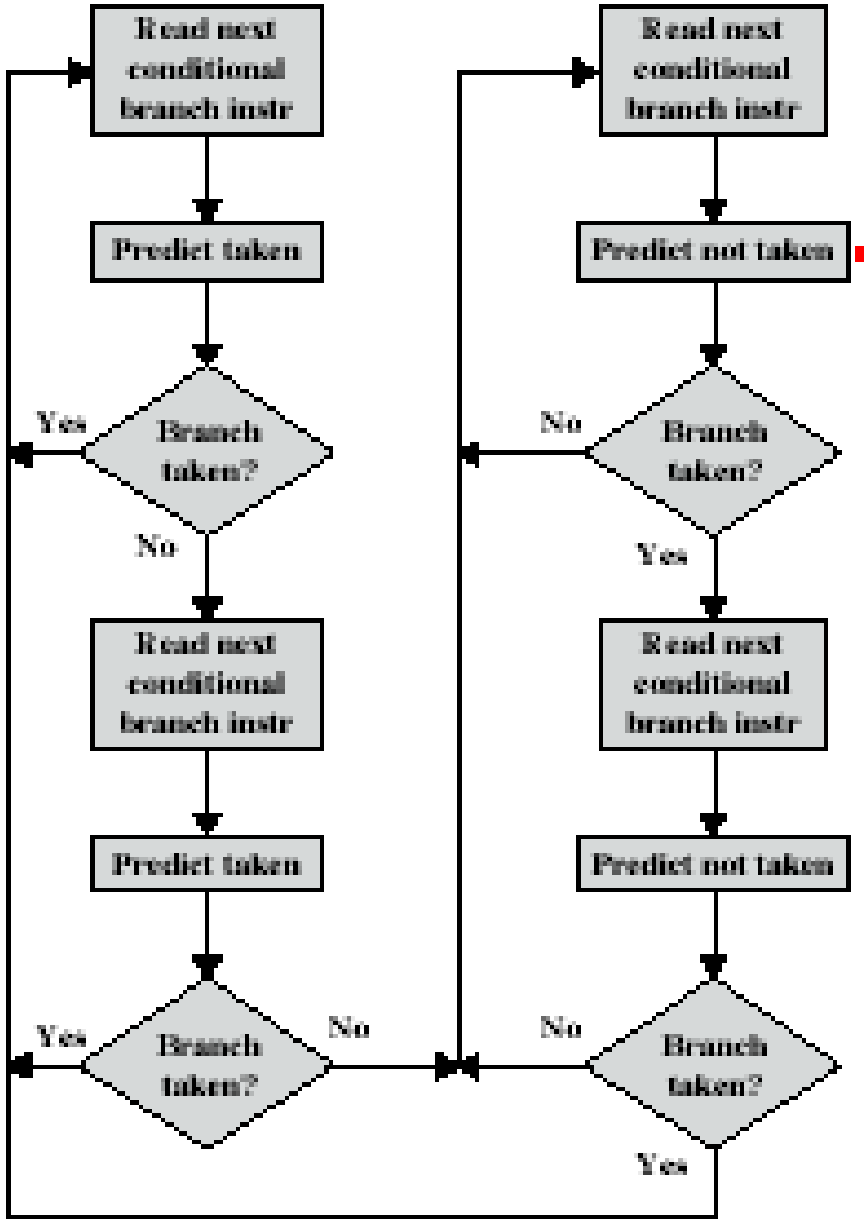
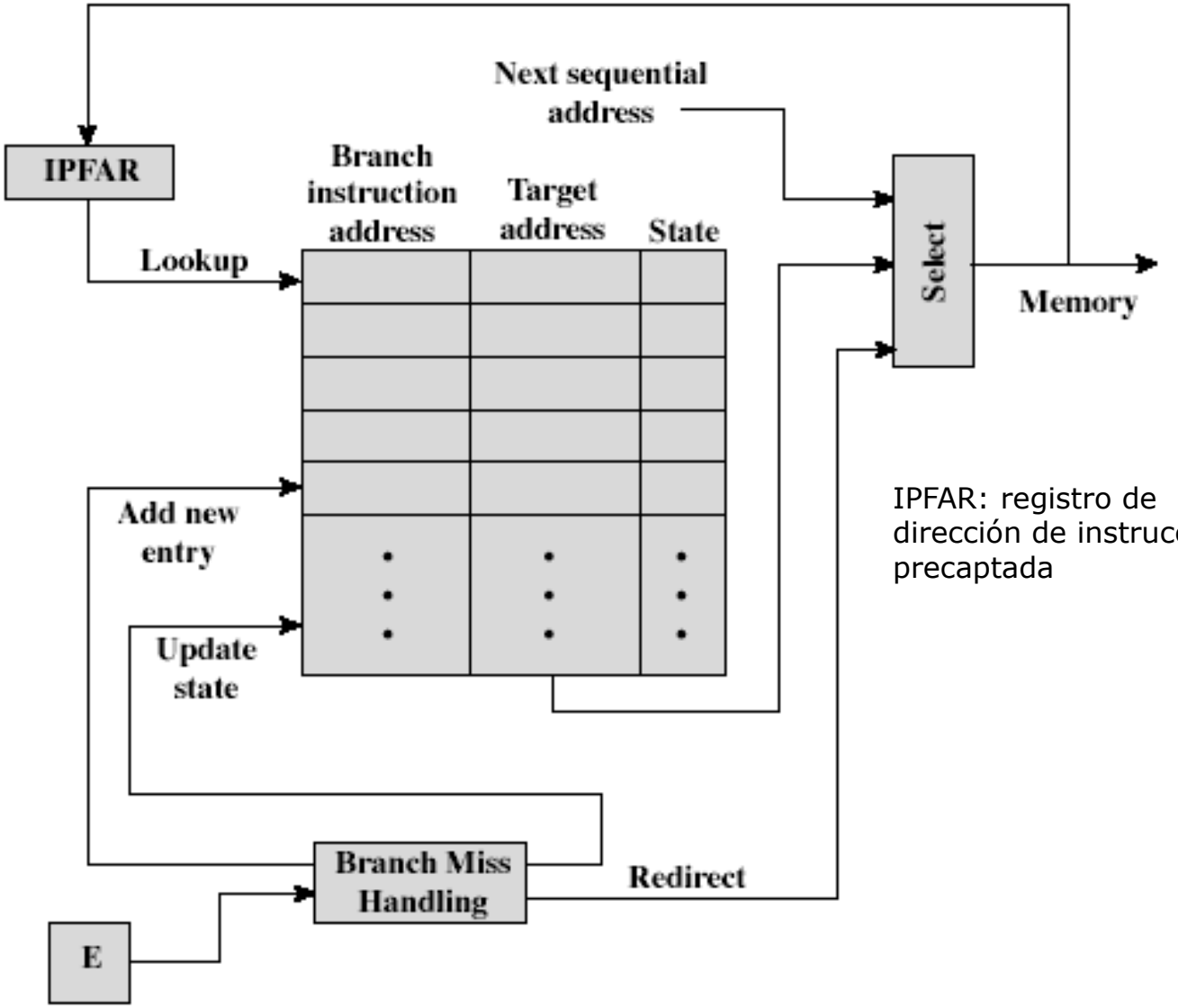


Tabla de historia de saltos (BTB)



IPFAR: registro de dirección de instrucción precaptada

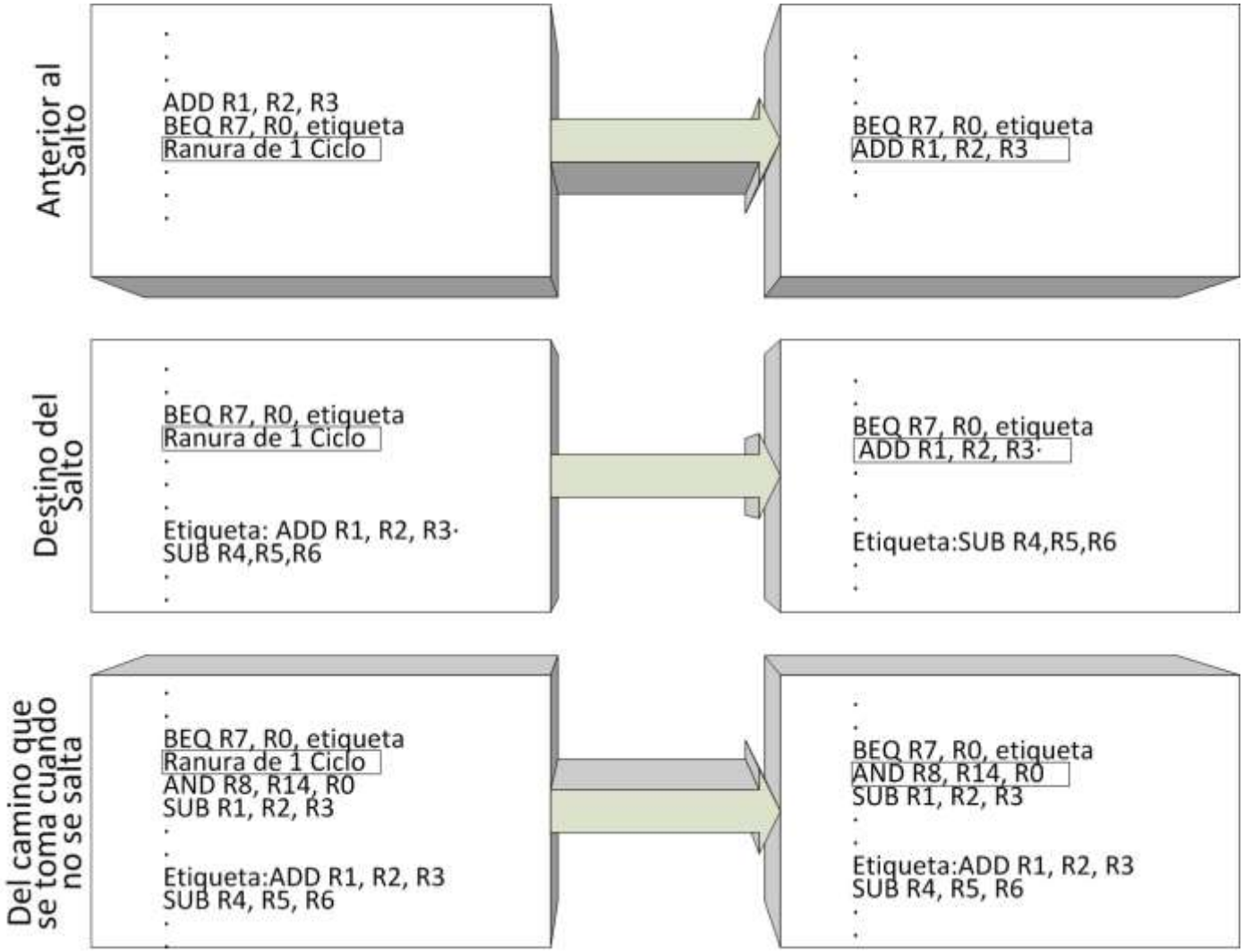
Salto retardado

Idea: Realizar trabajo útil mientras el salto se resuelve.

- *Hueco o ranura de retardo de salto* (delay-slot) es el período de penalización o parada luego de una instrucción de salto.
- El compilador trata de situar instrucciones útiles (que no dependan del salto) en los huecos de retardo. Si no es posible, se utilizan instrucciones NOP.
- Las instrucciones en los *huecos de retardo de salto* se captan siempre.

- Requiere reordenar las instrucciones.

Alternativas para el Salto retardado



Soluciones a riesgos de control (4)

- Otras soluciones hardware
  - Predecir según el código de operación
    - Hay instrucciones con más probabilidades de saltar
    - La tasa de acierto puede llegar a alcanzar un 75%
  - Flujos múltiples
  - Precaptar el destino del salto
  - Buffer de bucles

Flujos múltiples

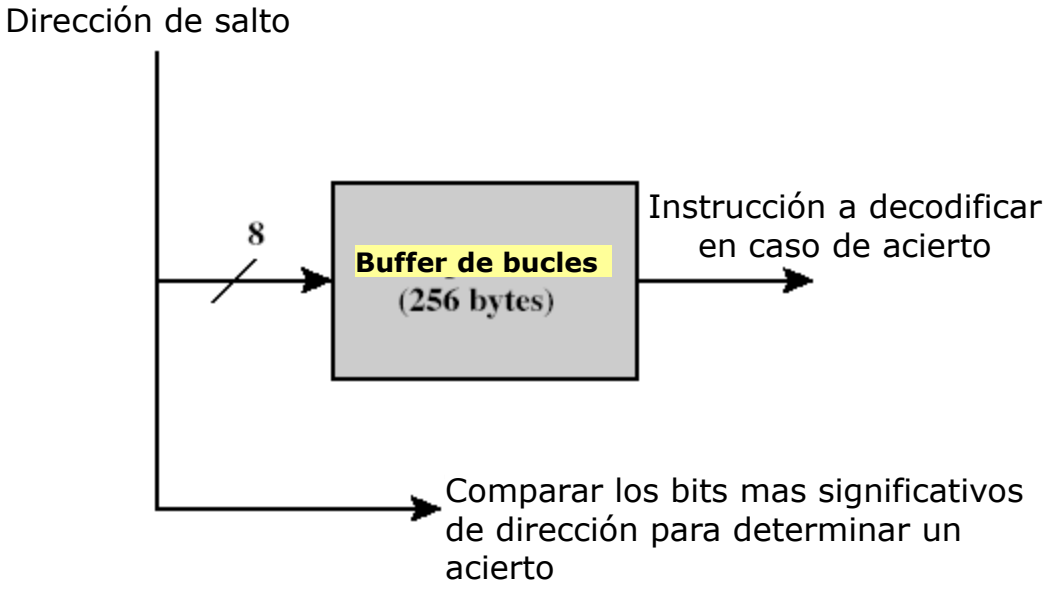
- Varios cauces (uno por cada opción de salto).
- Precaptan cada salto en diferentes cauces.
- Se debe utilizar el cauce correcto.
- Desventajas:
  - Provoca retardos en el acceso al bus y a los registros.
  - Si hay múltiples saltos, se necesita un mayor número de cauces.

Precaptación del destino de salto

- Se precapta la instrucción destino del salto, además de las instrucciones siguientes a la bifurcación.
- La instrucción se guarda hasta que se ejecute la instrucción de bifurcación.
  - El IBM 360/91 usa este método.

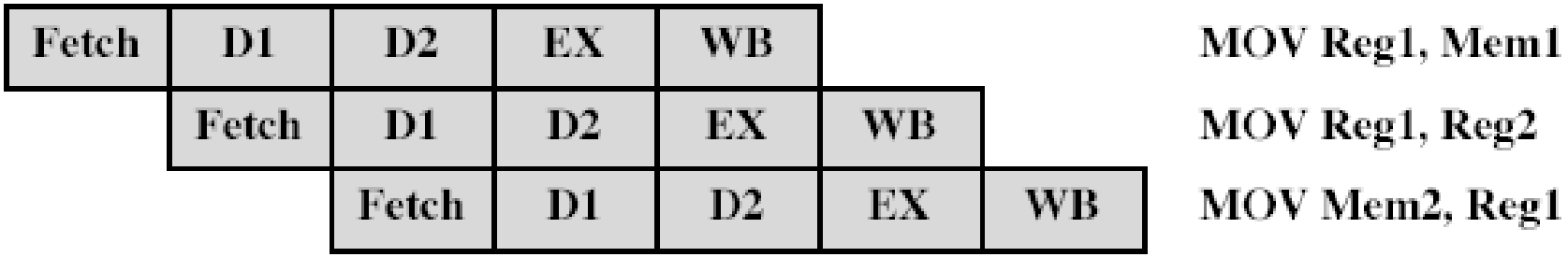
Buffer de bucles

- Memoria muy rápida. Gestionada por la etapa de captación de instrucción del cauce.
- Comprueba el buffer antes de hacer la captación de memoria.
- Muy eficaz para pequeños bucles y saltos.



Segmentación en el 80486

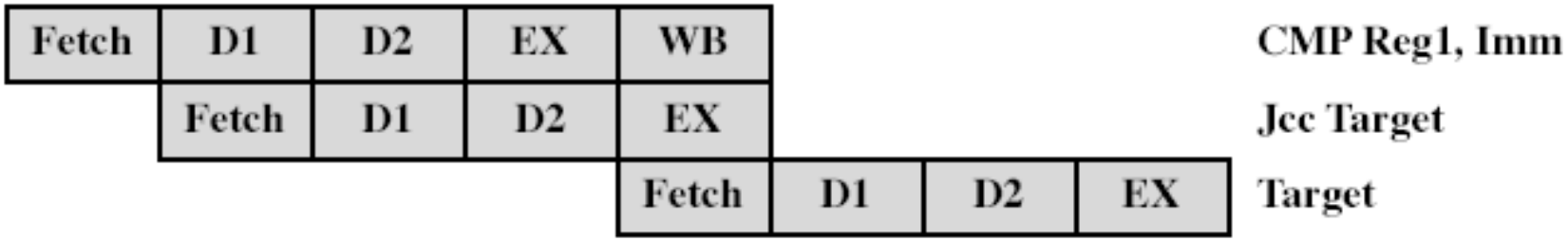
- Tiene un cauce de 5 etapas.
  - FI, D1, D2, EX y WB
- Ejemplos de funcionamiento del cauce



No hay retardo en el cauce por carga de un dato

Segmentación en el 80486 (2)

Retardo en el cauce por carga que utiliza un puntero



Temporizado por una instrucción de bifurcación

Mejoras al repertorio

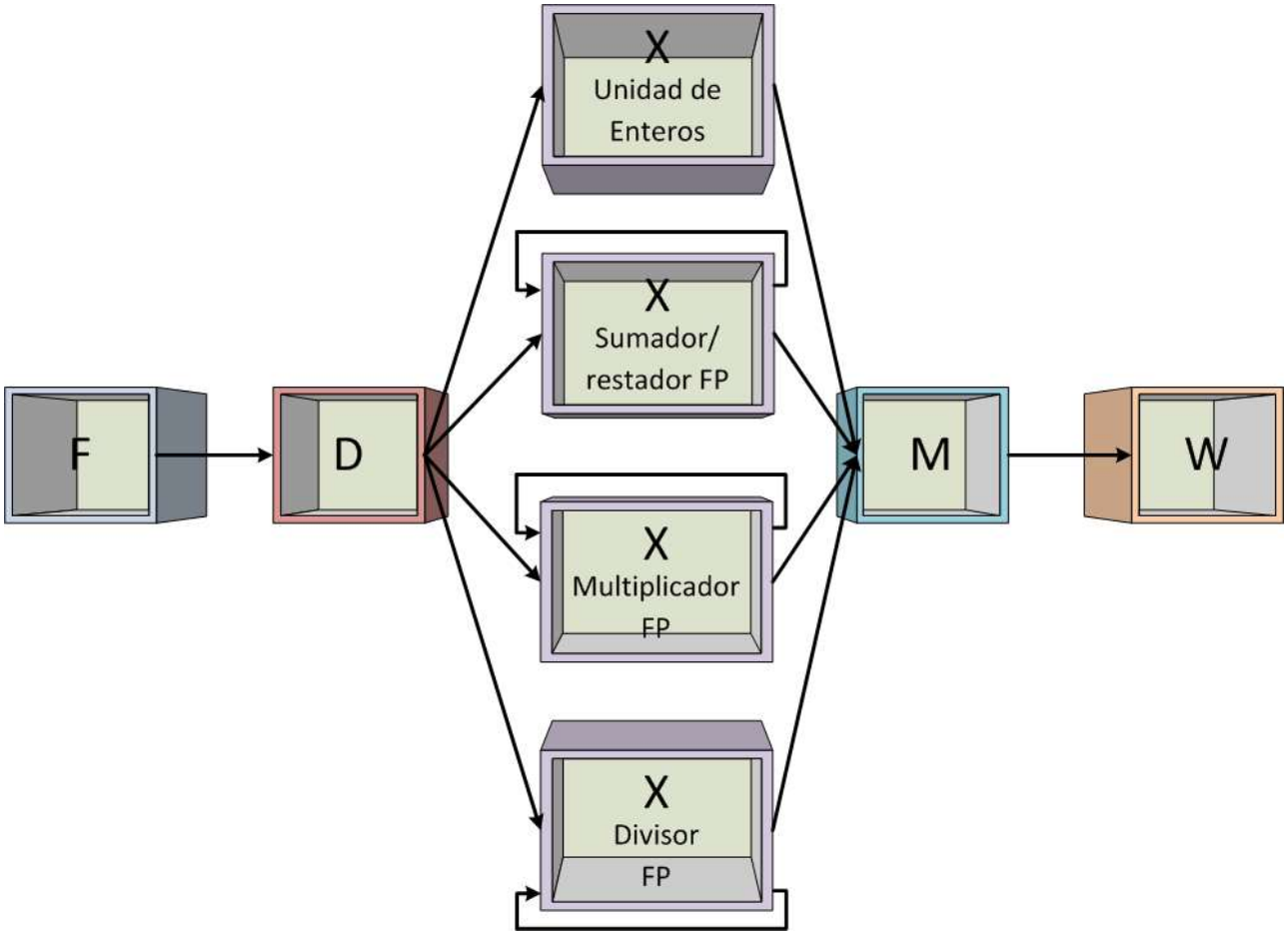
Quiero datos con representación en coma flotante

Necesito:

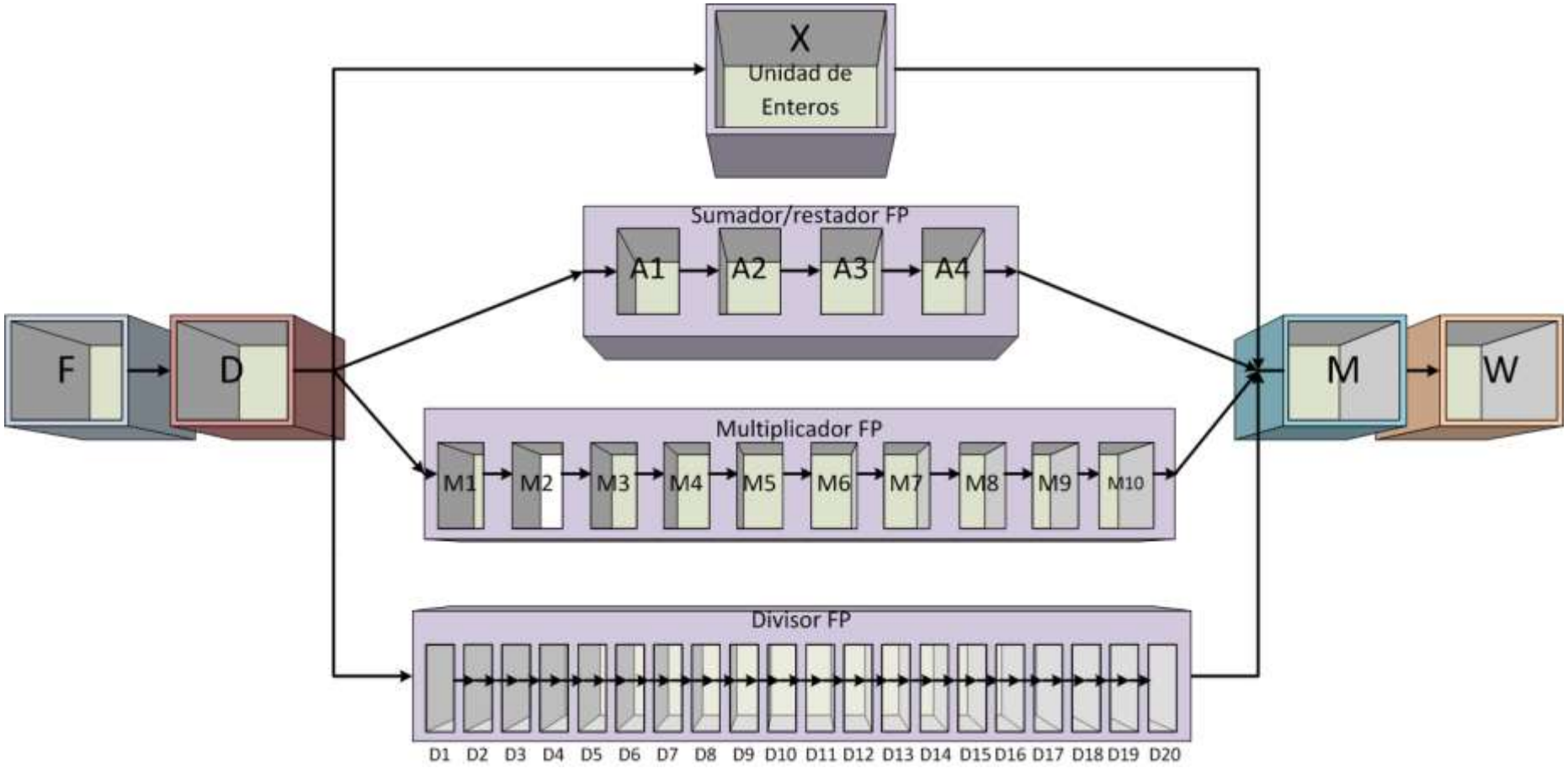
- Registros para su almacenamiento
- Hardware para las operaciones aritméticas
  - Suma y Resta
  - Multiplicación
  - División

¿Como serán las etapas del cauce de instrucciones?

Etapa de ejecución multifuncional



Unidades de ejecución segmentadas



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.