MICROARQUITECTURAS

Microarquitecturas e Introducción a Softcores

Contenido

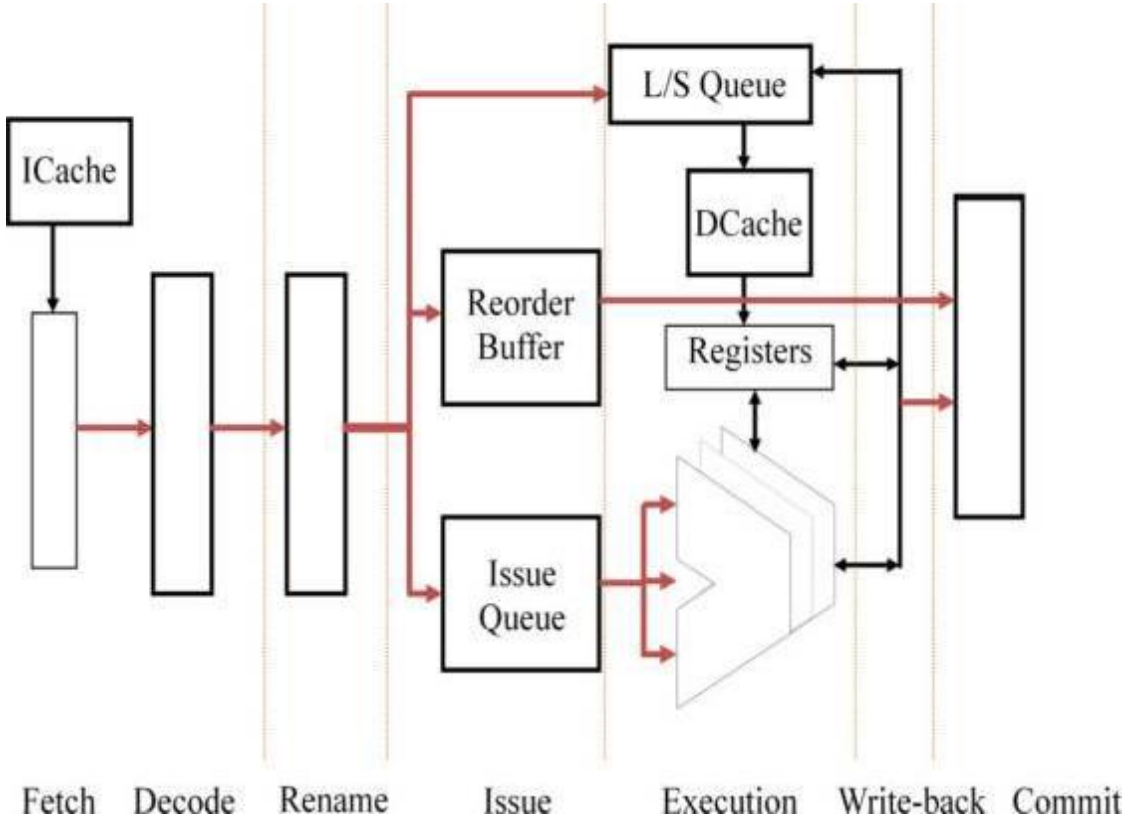
- Introducción
- Etapas de Procesamiento de una Instrucción
- Cache
- Obtención (Fetch)
- Decodificación (Decode)
- Renombrado (Rename)
- Asignación (Issue)
- Ejecución (Execute)
- Remisión (Commit)
- Bibliografía

INTRODUCCIÓN

- CISC vs RISC
- Clasificaciones
  - Pipelined / Nonpipelined
  - In-Order / Out-of-Order
  - Scalar / Superescalar
  - Vector Processors
  - Multicore Processors
  - Multithreaded Processors

Esquema Simplificado

- Fases por las que pasa una instrucción para su ejecución



CACHE

- Suele haber varios niveles (de 1 a 3)
- El nivel 1 (L1) es local al procesador y hay caches separados para instrucciones y datos
- Los niveles L2 y L3 son compartidos por los procesadores y estan juntos instrucciones y datos
- Las direcciones de un programa están en un espacio virtual, es necesario trasladarlas a las direcciones físicas correspondientes.

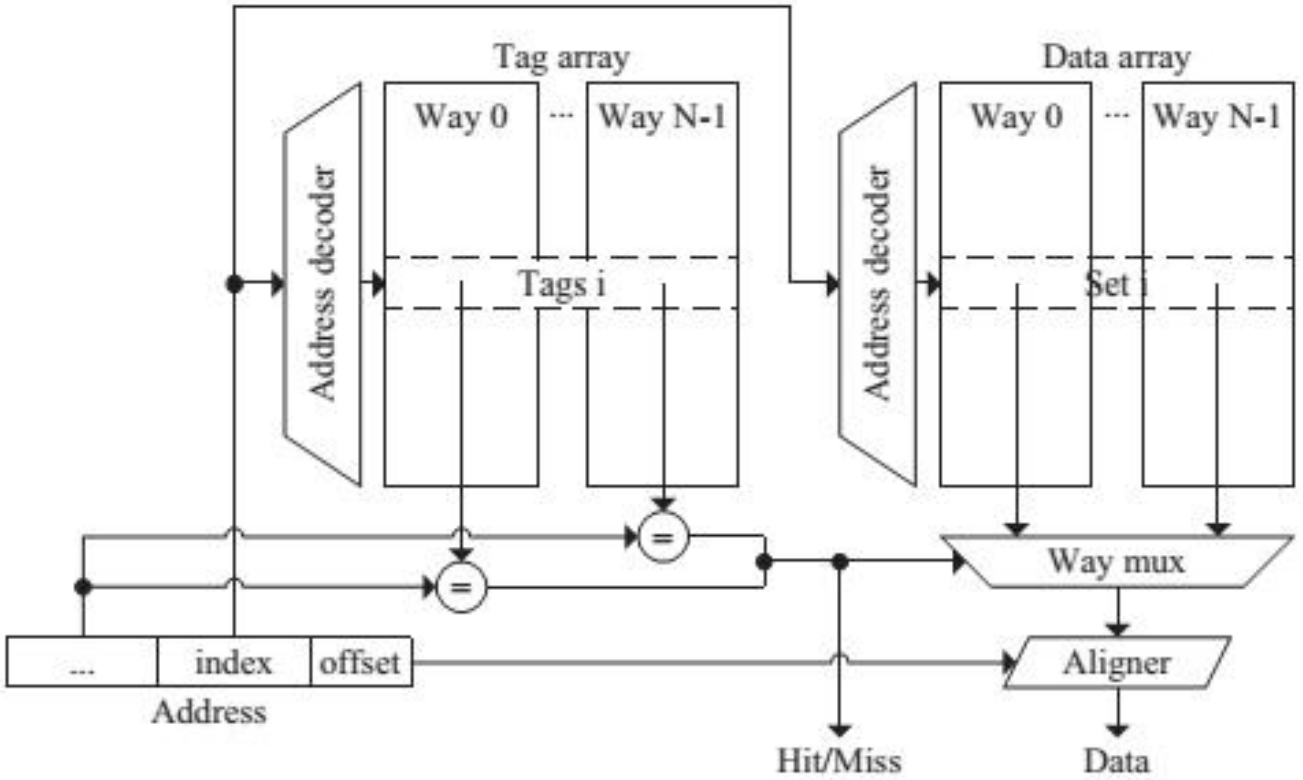
TRADUCCIÓN DE DIRECCIONES

- Espacio de memoria físico: aquel que puede generar el procesador en su bus de direcciones
- Espacio de memoria virtual: rango de direcciones disponibles para el programa.
- Ventajas del espacio de memoria virtual:
  - El direccionamiento dentro del programa es independiente de la memoria física del procesador
  - Se pueden establecer esquemas de protección del acceso a memoria por parte de los programas.
- La virtualización del espacio de memoria se realiza mediante “Paginado”, el cual puede ser administrado por el Procesador, por el S.O; o por ambos.
- Es posible que dos paginas virtuales se mapeen a la misma pagina física, esto se denomina “aliasing”, el procesador y el S.O. gestionan esta situación.

TRADUCCION DE DIRECCIONES

- El paginado se implementa mediante tablas con la equivalencia entre direcciones virtuales y físicas.
- Estas tablas se denominan “Translation Lookaside Buffers” (TLB). Incluyen información de la memoria (si es de codigo o datos, derechos de acceso, etc)
- En algunos procesadores las TLB son administradas por el S.O.
- En procesadores Intel las TLB son administradas por el hardware y casi totalmente transparentes al S.O.

ESTRUCTURA DEL CACHE



ESTRUCTURA DEL CACHE

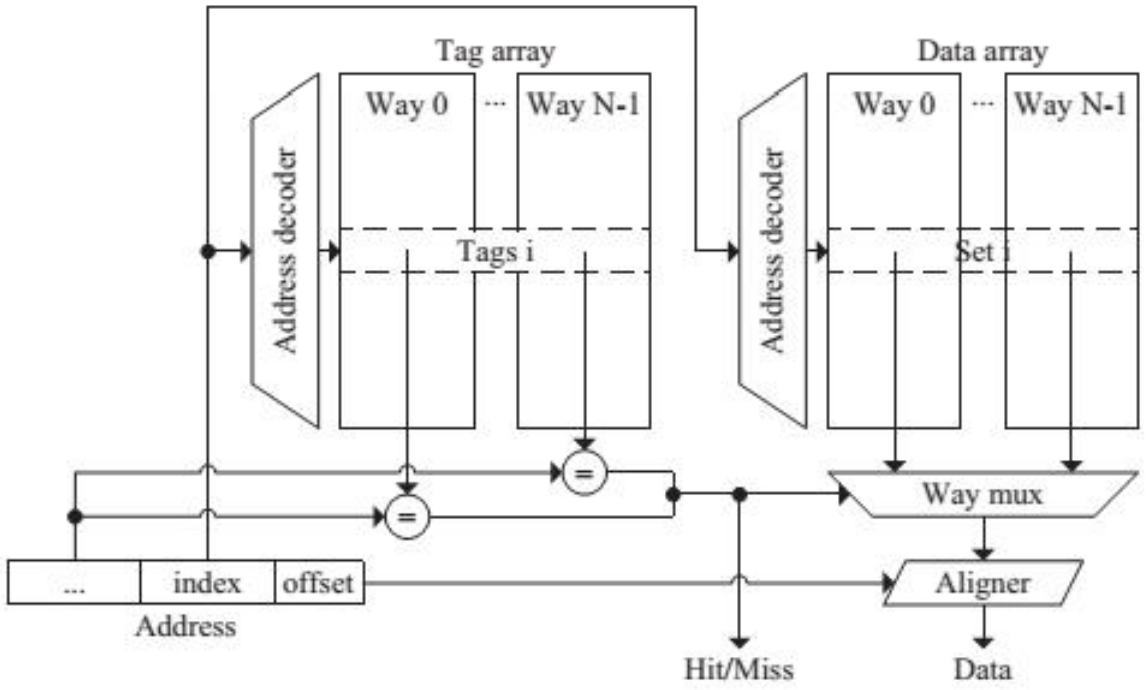
- Consiste en dos bloques, el Data Array y el Tag Array
- El Data Array se organiza en Sets. Cada Set es un grupo de bloques (blocks). La cantidad de bloques de un Set es el grado de asociatividad del cache. Si el Set tiene un solo bloque, el cache se denomina “de mapeo directo”.
- La memoria física se divide en tantas zonas como Sets tenga el cache.

ESTRUCTURA DEL CACHE

- Como el Set representa una zona de memoria física, pero el bloque almacena una cantidad de bytes mucho menor que la zona que representa (en general 64 bytes), una mayor asociatividad aumenta la probabilidad de hit, ya que se pueden tener distintas subzonas (bloques) asociadas a una zona (Set).
- Diferentes direcciones pueden corresponder al mismo Set. El Tag Array resuelve a que bloque del set corresponde la solicitud.

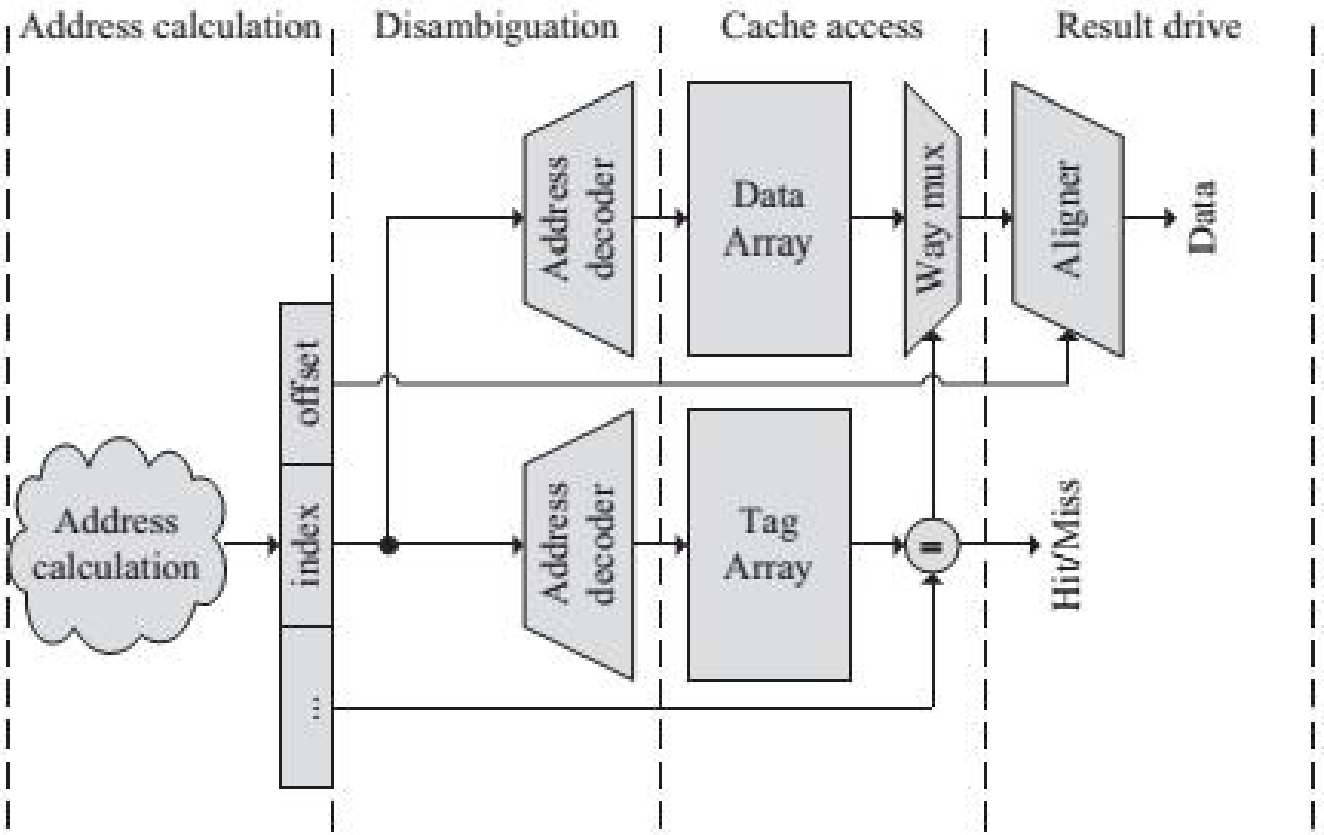
ESTRUCTURA DEL CACHE

- El acceso al Tag y Data Array puede ser en paralelo o en serie.
- Estructura del cache de acceso Paralelo



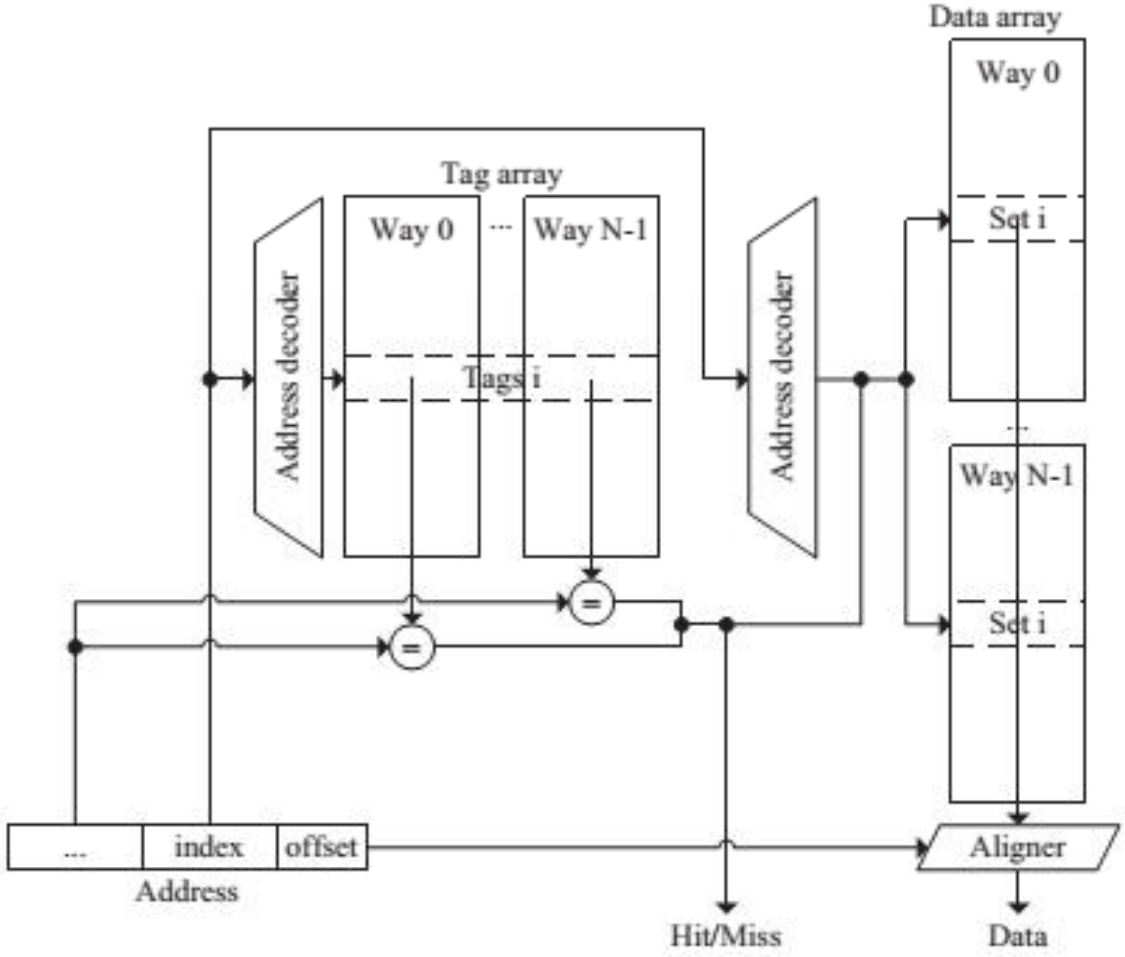
ESTRUCTURA DEL CACHE

- Secuencia de acceso en paralelo (4 etapas)



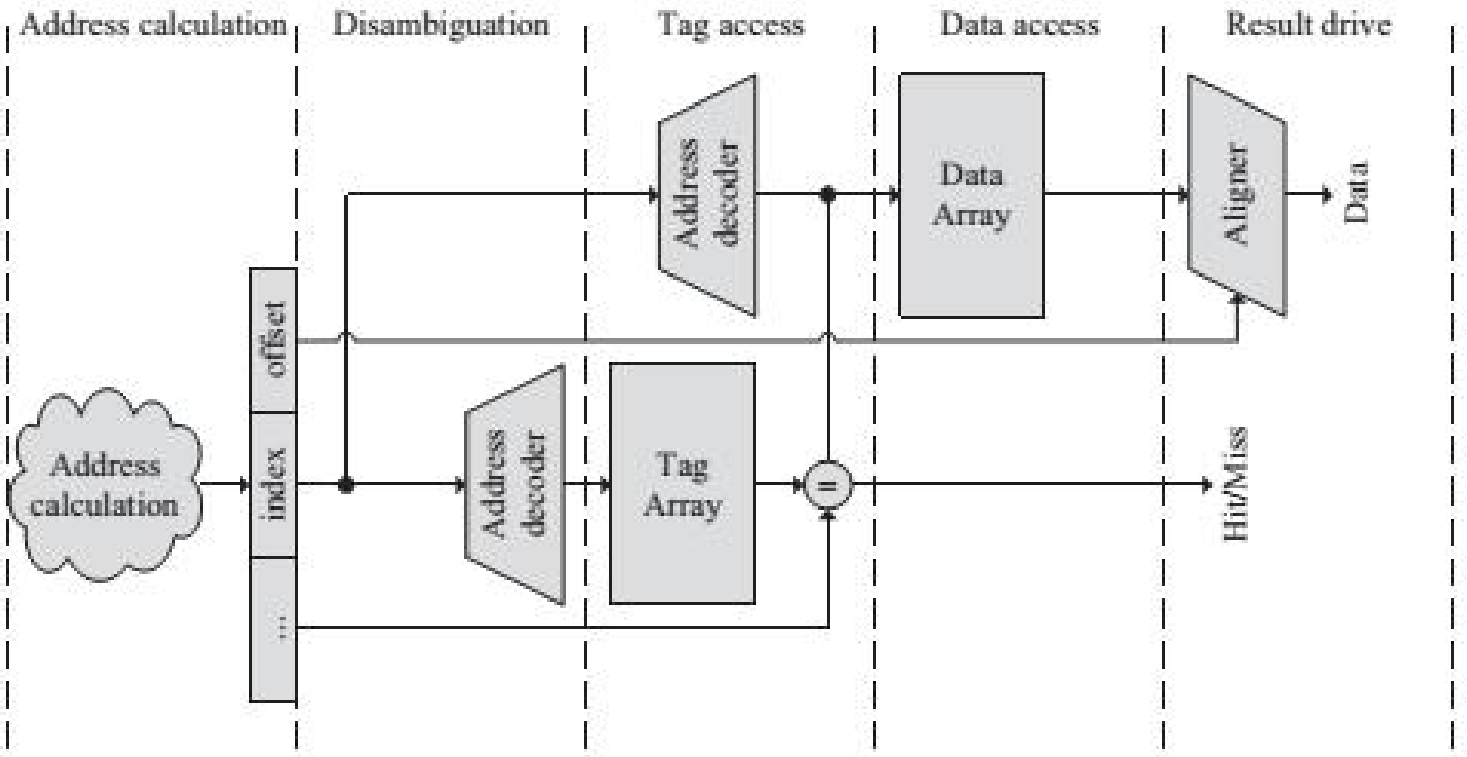
ESTRUCTURA DEL CACHE

- Estructura del cache de acceso en Serie



ESTRUCTURA DEL CACHE

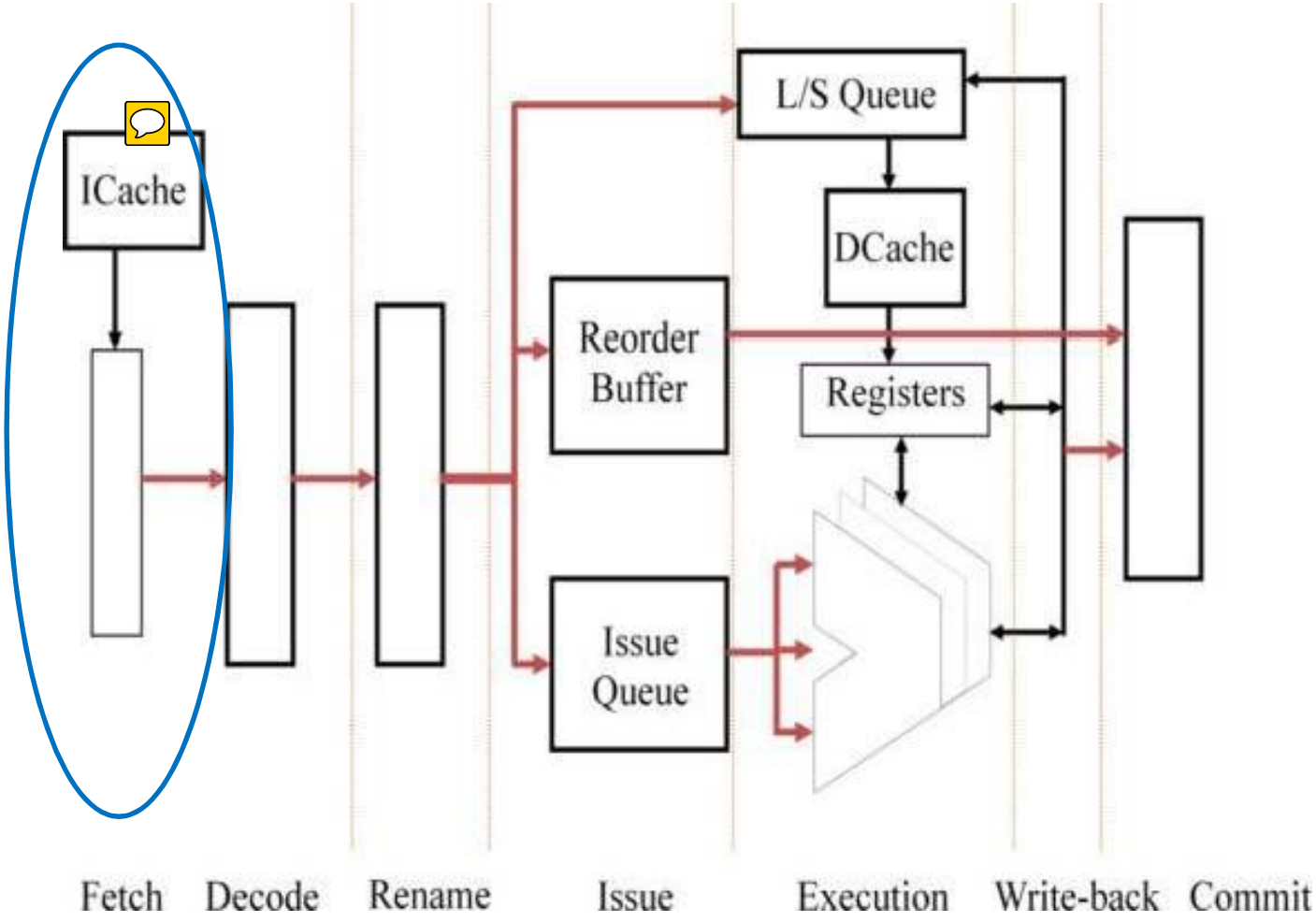
- Secuencia de acceso en serie (5 etapas)



ESTRUCTURA DEL CACHE (COMPROMISOS)

- En el cache de acceso serie no esta el multiplexor de bloques y por ende consume menos al ser mas simple, pero requiere un ciclo mas para resolver el acceso
- Los procesadores out-of-order, como enmascaran las latencias de acceso a memoria, suele utilizar acceso serie.
- Los procesadores in-order, donde la latencia de acceso a memoria es importante, suelen utilizar acceso paralelo.
- Los caches de mapeo directo son los mas rapidos, pero hay alta posibilidad de fallo.
- En los caches asociativos, cuanto mas asociatividad, menor probabilidad de fallos, pero mayor complejidad del multiplexor del Tag Array

Etapa de Obtención (fetch)



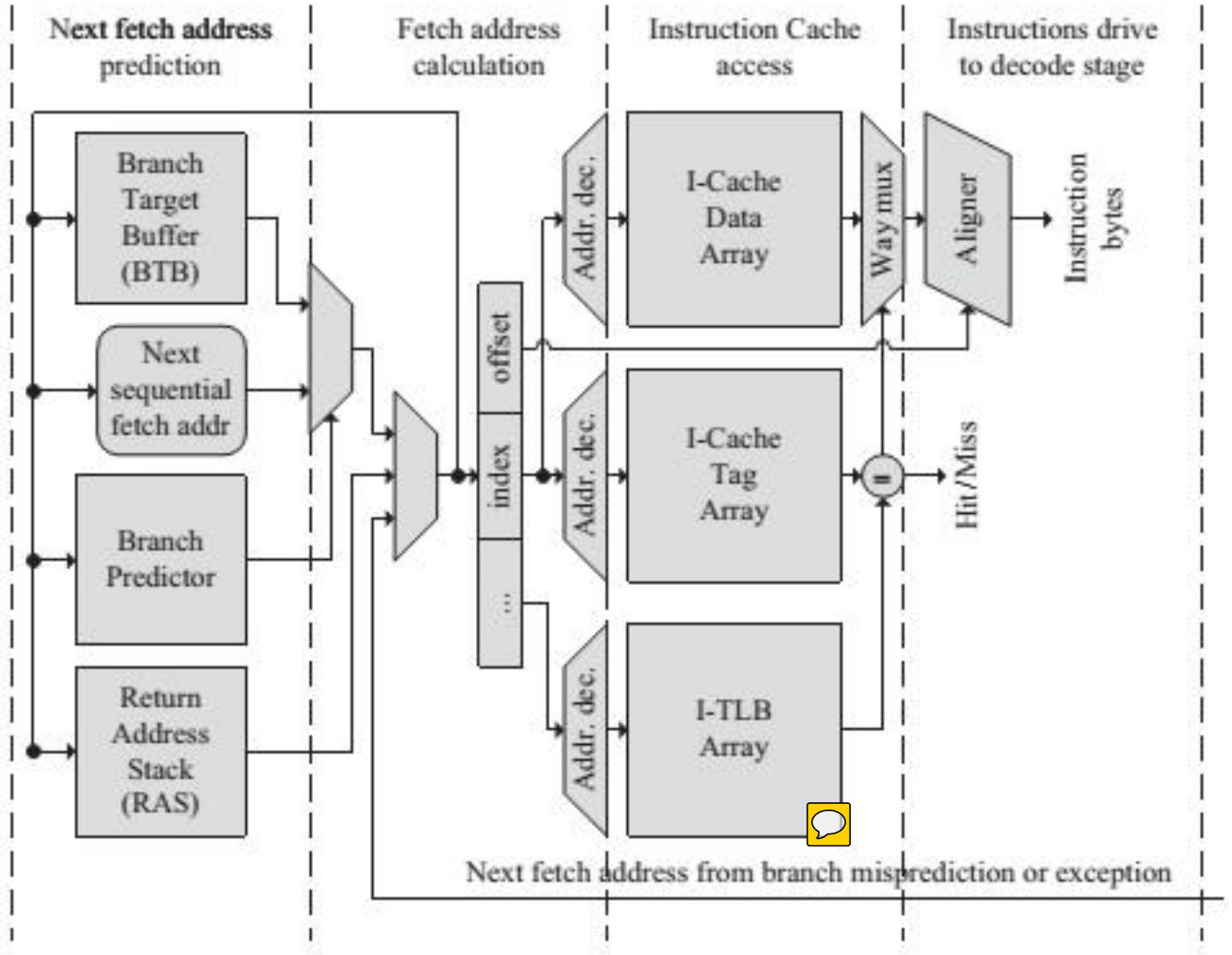
Etapa de obtención: Unidad de obtención de instrucciones

- Su función es proveer al procesador con instrucciones.
- Consiste en un cache de instrucciones y la lógica para calcular la dirección de la siguiente instrucción.
- Se obtiene una instrucción por ciclo, por lo que el calculo de la siguiente dirección se debe hacer en paralelo con el acceso al cache de instrucciones.
- En las bifurcaciones (branch) se debe predecir si se realiza o no la bifurcacion y la dirección de la siguiente instrucción de la bifurcación.

Unidad de obtención de instrucciones

- La predicción de si se realiza o no la bifurcación la realiza la unidad de predicción de bifurcación (Branch Predictor)
- El calculo de la siguiente dirección de la bifurcación la realiza el bufer de destino de bifurcación (Branch Target Buffer – BTB)
- Algunos procesadores consideran el retorno de subrutina como un caso particular de bifurcación y usan un stack de retorno (Return Address Stack – RAS) para predecir la siguiente dirección.

Unidad de obtención de instrucciones

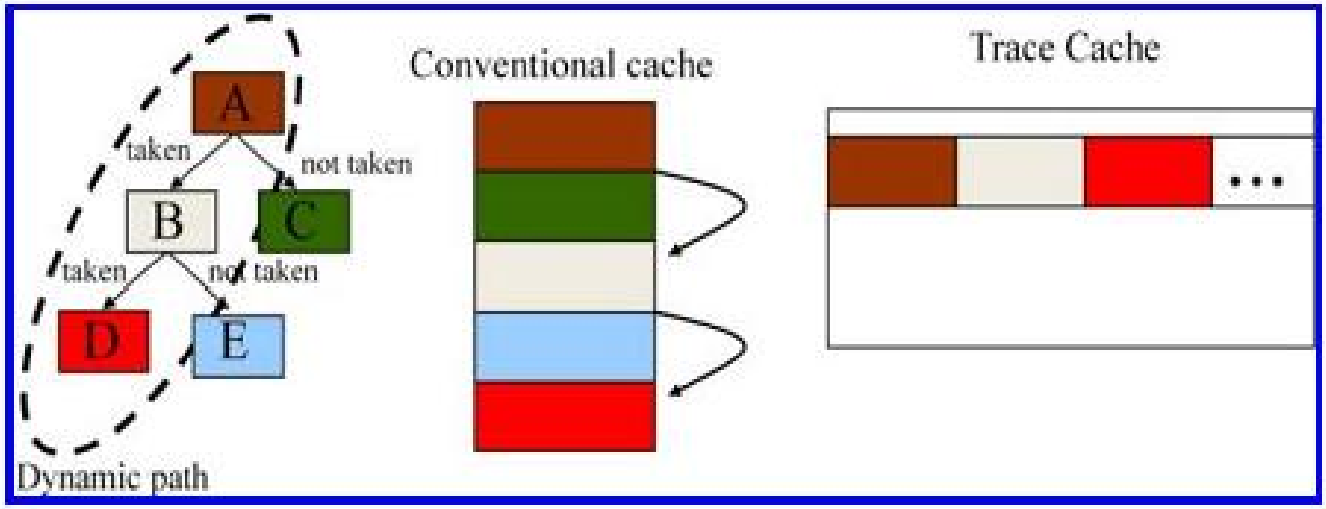


UNIDAD DE OBTENCIÓN DE INSTRUCCIONES – CACHE DE INSTRUCCIONES

- Guarda instrucciones que es probable que sean accedidas.
- Tienen capacidad de decenas de KB en bloques de 64 bytes.
- Los procesadores superescalares realizan accesos a posiciones consecutivas del cache.
- Los caches convencionales almacenan las instrucciones en el orden en que aparecen en el ejecutable (orden estatico)
- Los caches con trazabilidad (Trace Cache) almacenan las instrucciones en forma dinamica.

UNIDAD DE OBTENCIÓN DE INSTRUCCIONES – CACHE DE INSTRUCCIONES

- Diagrama



- En cache convencional cada instrucción aparece solo una vez, en Trace cache puede aparecer mas veces (si la misma instrucción esta en distintos caminos).
- El cache dinámico tiene mejor performance frente a branch.

UNIDAD DE OBTENCIÓN DE INSTRUCCIONES – BRANCH TARGET BUFFER

- El caso mas común de bifurcación es ser relativa a la posición actual, siempre a la misma dirección, y no muy lejos de la posición actual.
- Para estos casos, en vez de predecirse la siguiente dirección, se almacenan las direcciones de destino de la bifurcación en una tabla denominada “Branch Target Buffer” (BTB).
- Los valores de la BTB suelen ser aplicables tambien para bifurcaciones cuyo destino se calcula al momento de procesar la instrucción de bifurcación.

Unidad de obtención de instrucciones – return address stack

- El retorno de subrutina es un caso particular de bifurcación.
- Para estos casos, en vez de predecirse la siguiente dirección, se almacenan las direcciones de retorno en una memoria LIFO denominada "Return Address Stack" (RAS).

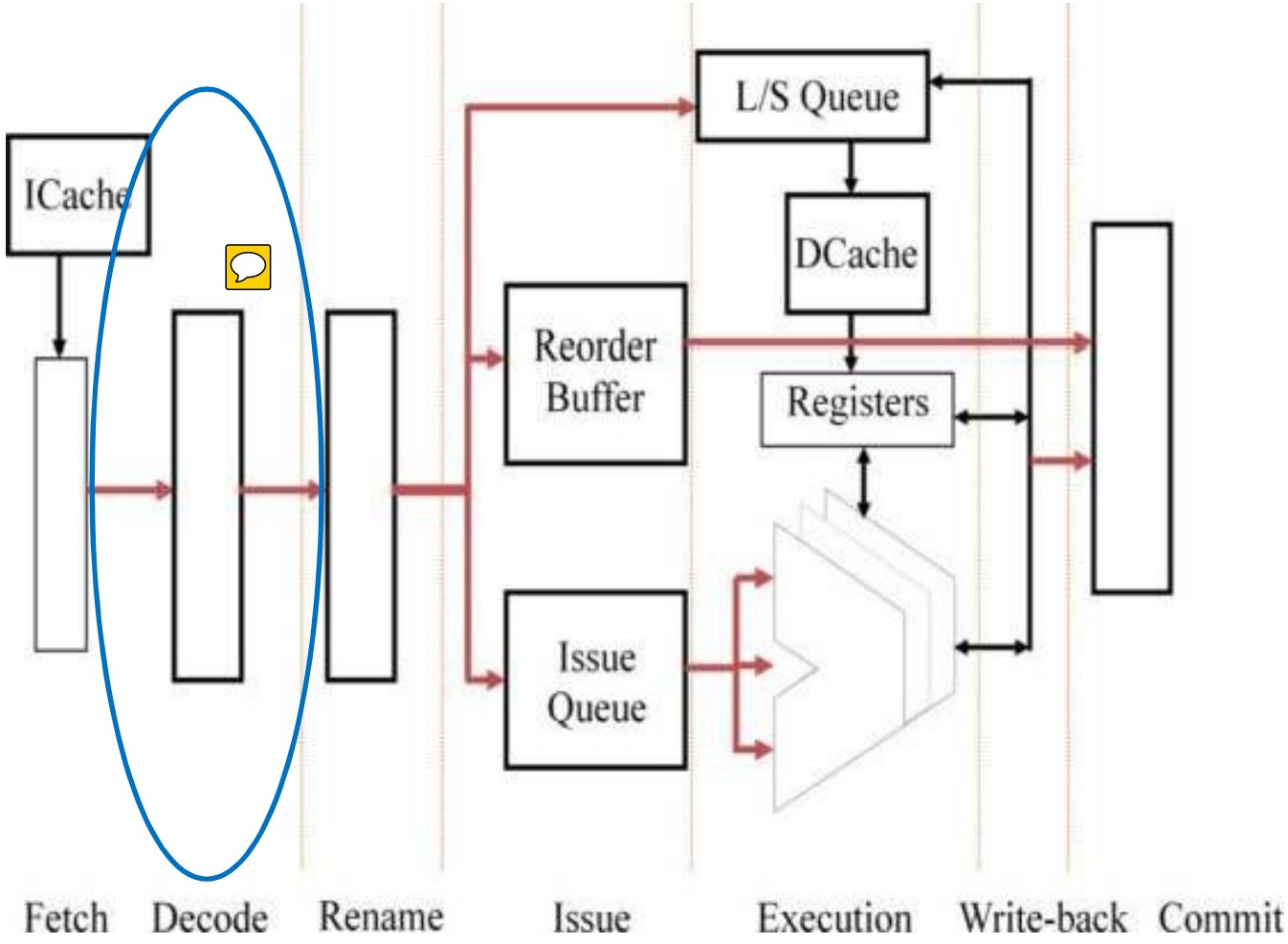
Unidad de obtención de instrucciones – predicción en bifurcaciones condicionales

- La bifurcación condicional es un problema porque solo se sabe si se tomo la bifurcación en la etapa de ejecución de la instrucción.
- Puede haber 10 o mas ciclos entre la etapa de Obtención y la de Ejecución, por lo que es necesario poder predecir la bifurcación.
- La predicción es de dos tipos: Estática y Dinámica

UNIDAD DE OBTENCION DE INSTRUCCIONES – PREDICCION EN BIFURCACIONES CONDICIONALES

- Predicción Estática:
  - Se realiza en base a profilling. Dado lo ocurrido en las ultimas N veces, se asume que la bifurcación va a ocurrir o no.
  - Es simple de implementar.
- Predicción Dinámica:
  - Se realiza en base a estadísticas de ejecución y en base a la probabilidad de ocurrencia, se decide en cada caso particular si la bifurcación va a ocurrir o no.
  - Hay distintos algoritmos de predicción dinámica, implementados mediante maquinas de estado.

Etapa de Decodificación (decode)

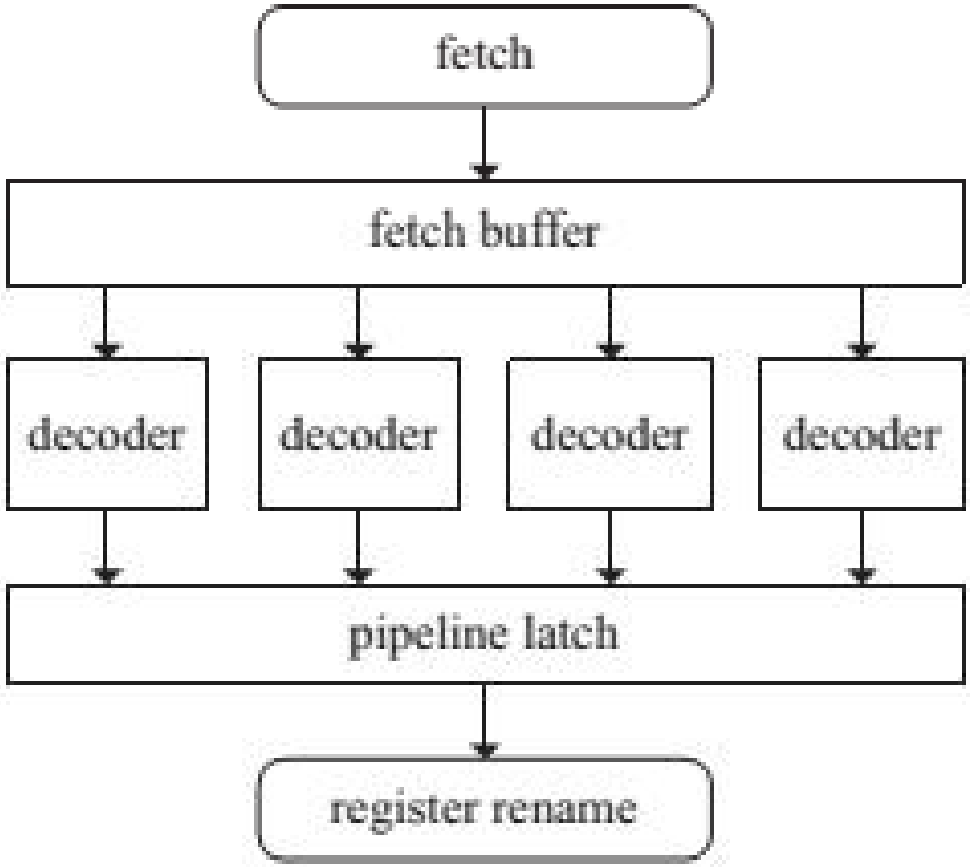


Etapa de Decodificación

- En esta etapa se analiza la semántica de la instrucción y se define como debe ser ejecutada. Se identifica:
  - El tipo de instrucción (control, acceso a memoria, aritmética, etc)
  - El tipo de operación de la instrucción; por ejemplo, si es aritmética, que operación de la ALU se utilizara, si es de bifurcación, que condición verificar (igualdad, mayor, etc)
  - Que recursos requiere la instrucción (registros de entrada y de salida)

Etapa de Decodificación

- Estructura del decodificador

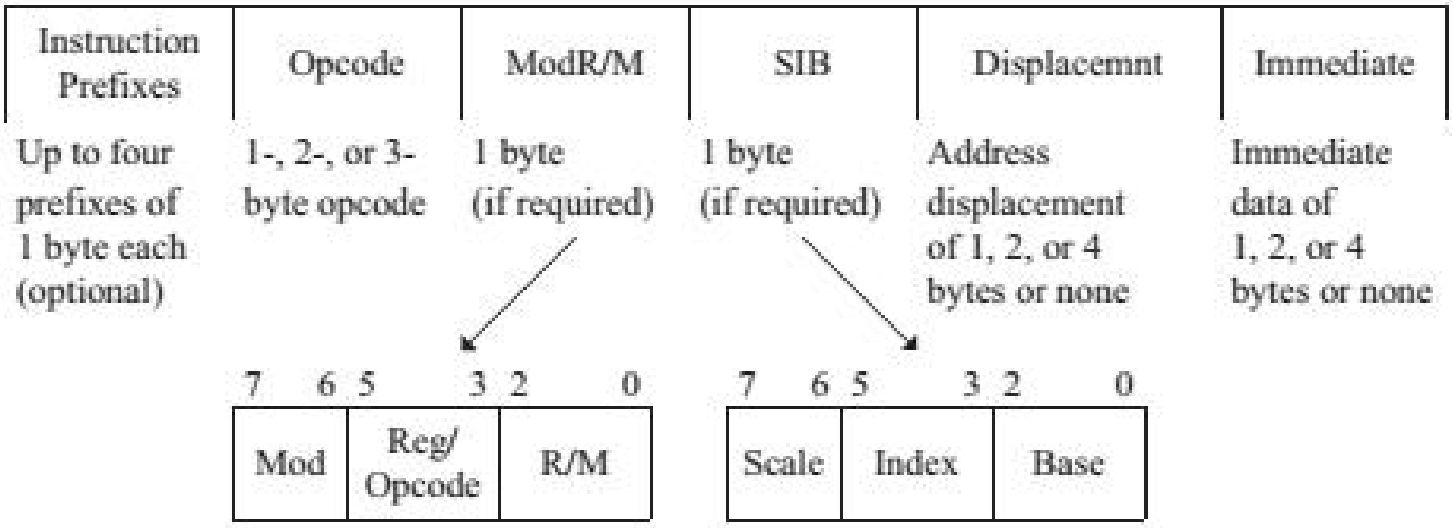


Etapa de Decodificación

- Instrucciones RISC
  - Tienen tamaño fijo
  - Pocas variantes de codificación (pocas variaciones de posición del opcode y los operandos)
  - Permiten decodificación en 1 ciclo
- Instrucciones CISC
  - Tienen estructura y longitud variable
  - Necesita una interpretación secuencial (el contenido de algunos campos modifica la interpretación y longitud de otros)
  - La decodificación lleva varios ciclos

ETAPA DE DECODIFICACIÓN

- Formato de instrucción CISC (x86)

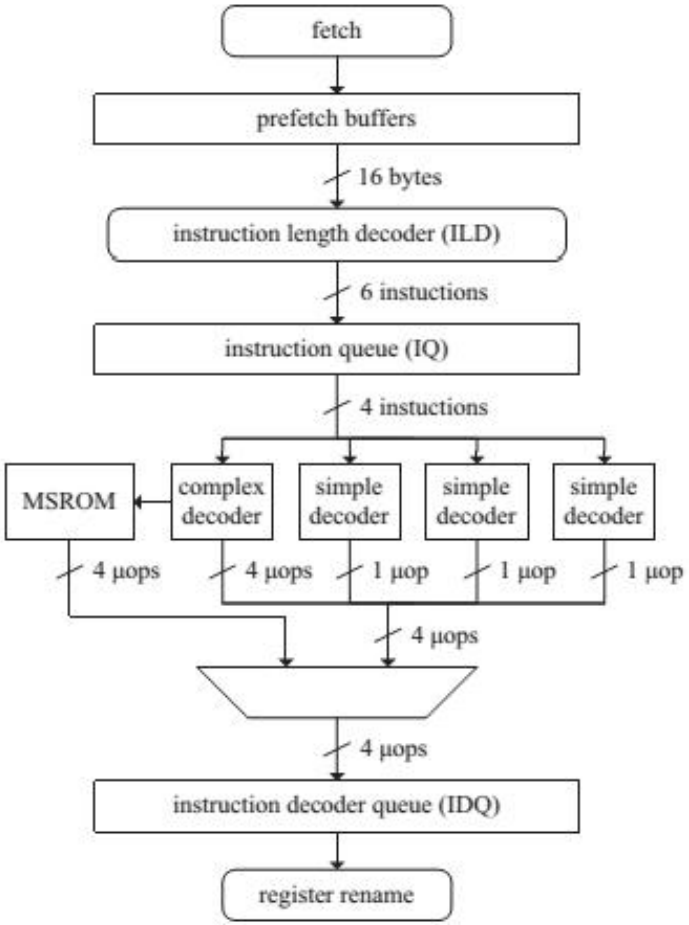


Etapa de Decodificación

- Las instrucciones CISC se convierten dinamicamente en una secuencia de instrucciones RISC, denominadas "microoperaciones" (uops)
- Las uops tienen las características de las instrucciones RISC
  - Longitud fija
  - Formato regular
  - Tres operandos (dos fuentes y 1 destino)

Etapa de Decodificación (Intel i7)

- Estructura



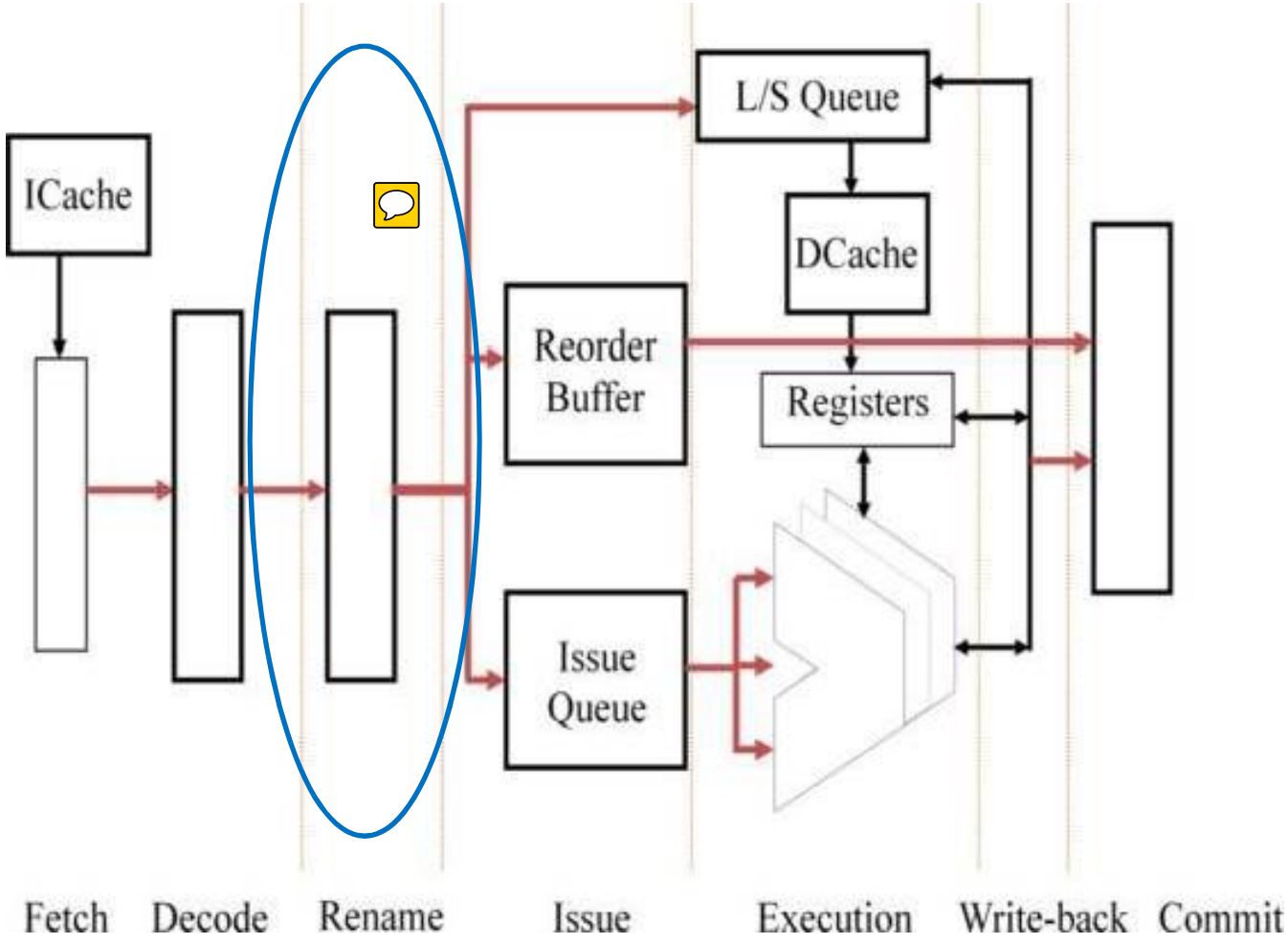
Etapa de Decodificación (Intel i7)

- La decodificacion se hace en dos etapas:
  - Instruction Length Decoder (ILD)
  - Dynamic Translation (DT)
- La ILD separa el stream de bytes del cache de instrucciones en secuencias de instrucciones x86
- La DT convierte las instrucciones x86 CISC en uops RISC
- Ambas etapas estan acopladas a traves de la Instruction Queue (IQ), que enmascara burbujas producidas por los diferentes tiempos de obtención de la secuencia de instrucciones x86

Etapa de Decodificación (Intel i7)

- Instruction Length Decoder:
  - Procesa en bloques de 16 bytes
  - Establece la longitud de la instrucción
- Dynamic Translator:
  - Las instrucciones que pueden convertirse a 1 uOp se procesan en los “simple decoder”
  - Las instrucciones que requieren una secuencia de hasta 4 uOps se procesan en el “complex decoder”
  - Las instrucciones que requieren mas de 4 uOps se procesan en el Microsequencer (MSROM), que genera la secuencia de uOps necesaria para implementar la instrucción.

Etapa de Renombrado (rename)



Etapa de Renombrado

- Se realizan dos funciones:
  - Reasignación de Registros
  - Despacho de las Instrucciones
- Reasignación de Registros
  - Elimina falsas dependencias
  - Es mas común en procesadores out-of-order
  - La dependencia puede ser de Datos o de Nombre
    - Dependencia de Datos: El resultado de la instrucción es usado por otra instrucción
    - Dependencia de Nombre: Un registro utilizado por la instrucción es utilizado también por otra instrucción. Esta dependencia puede ser de “escritura después de escritura” o “escritura después de lectura”

ETAPA DE RENOMBRADO

- Dependencias en las Instrucciones

r1 = r2 + r3	r1 = r2 + r3	r1 = r2 + r3
....	....	....
r4 = r1+r5	r1 = r4 + r5	r2 = r4 + r5
Data dependence	**Name dependence**	**Name dependence**
Read after write	**Write after Write**	**Write after read**

- El renombrado se puede hacer mediante tres esquemas:
  - Bufer de reordenamiento (Reorder Buffer – ROB)
  - Bufer de renombrado (Rename Buffer)
  - Unión de Registros (Merged Register File)

ETAPA DE RENOMBRADO

- Reorder Buffer
  - Los valores se guardan en dos lugares: en el ROB y en los Registros del procesador (Arquitectural Register File – ARF)
  - El ROB guarda los resultados de las instrucciones que no llegaron a la etapa de remisión (commit)
  - El ARF guarda el resultado generado por instrucciones que finalizaron (commited).
  - Hay una tabla que indica para cada registro del ARF si su valor actualizado esta en el ROB o en el ARF

Etapa de Renombrado

- Rename Buffer
  - Es una variación del esquema con ROB
  - Implementa una estructura independiente para almacenar los resultados de las instrucciones en proceso.
  - Solo las instrucciones que producen un resultado en registros consumen espacio de esta estructura, a diferencia del ROB, que es un duplicado del ARF, por lo que un resultado puede estar simultaneamente en el ROB y en el ARF, consumiendo espacio y haciendo mas complejo el renombrado.
  - Este esquema se utiliza en los PowerPC de IBM

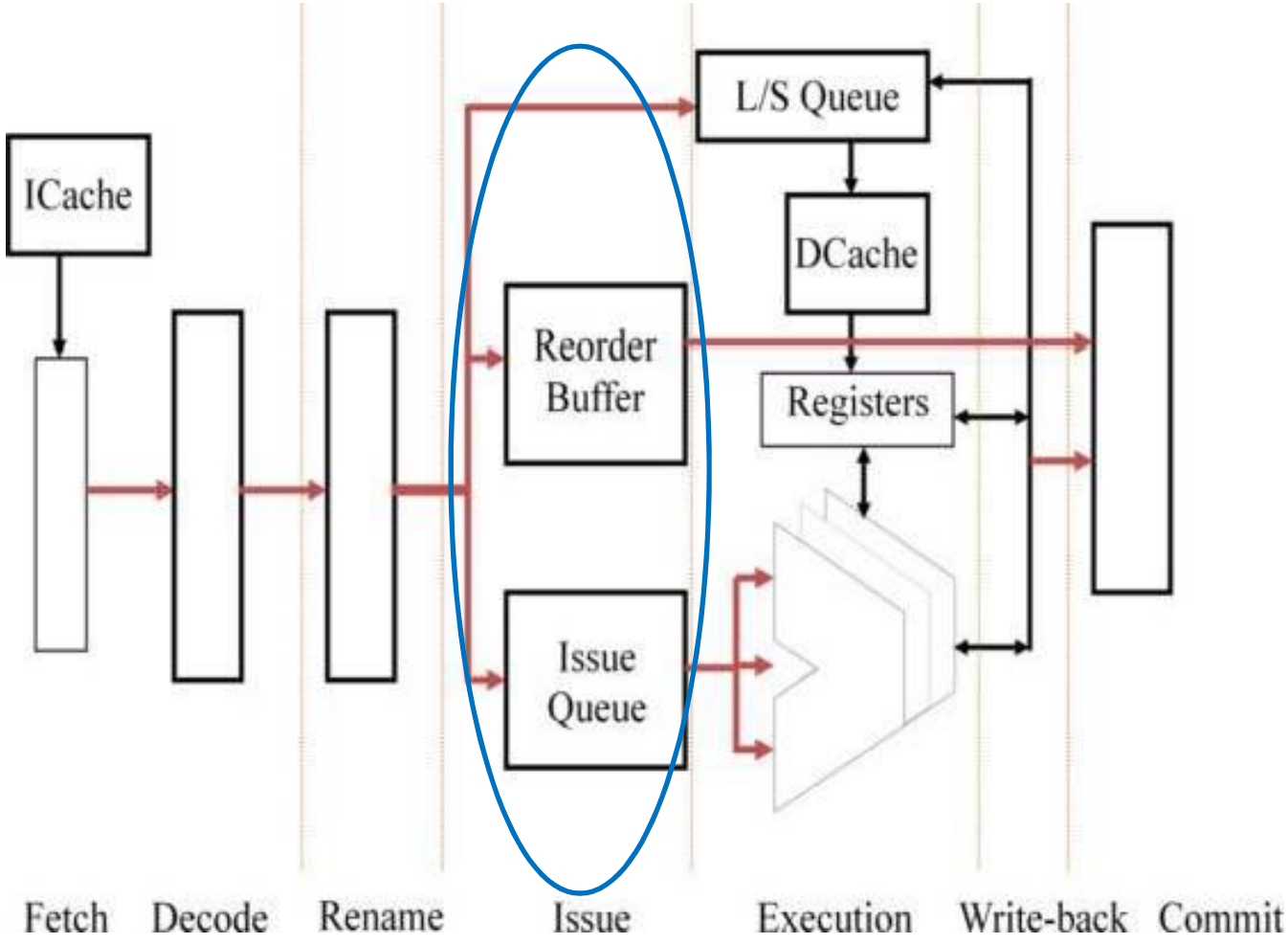
Etapa de Renombrado

- Merged Register Buffer
  - Hay un único grupo de registros que tiene los valores especulativos (instrucciones no finalizadas) y remitidos (instrucciones finalizadas)
  - Cada registro puede estar “libre” o “asignado”
  - Los registros libres estan disponibles para usarse
  - Los registros asignados estan en uso y pueden tener un resultado especulativo, un resultado remitido, o no tener un valor (el registro esta en uso pero aun no se genero su valor)
  - Este esquema se uso en el Pentium4

Etapa de Renombrado

- Comparación entre los esquemas
  - La implementación de ROB y Rename Buffer son las mas simples.
  - En el Merged Register File el registro se escribe una única vez, no hay traslado de valores ni duplicación.
  - En el Merged Register File los registros fuente provienen de una única ubicación, en ROB y Rename Buffer los datos pueden estar en dos lugares distintos y debe haber lógica para obtener el dato del lugar correcto.

Etapa de Asignación (issue)



Etapa de Asignación

- En esta etapa se asignan las instrucciones a las unidades funcionales correspondientes.
- En los procesadores in-order las instrucciones se asignan en el orden en que aparecen
- En los procesadores out-of-order las instrucciones se asignan a medida que los datos de entrada están disponibles

Etapa de Asignación

- Asignación en procesadores in-order
  - Se implementa mediante tablas (ScoreBoard)
  - Hay una tabla de dependencias, que almacena el estado de los registros; y una tabla de recursos, que almacena el estado de las unidades funcionales.
  - La tabla de dependencias se indiza de acuerdo a los registros que utiliza la instrucción, a su vez el estado de cada registro puede ser disponible o no disponible.
  - La tabla de recursos contiene información de las unidades funcionales disponibles
  - En los procesadores que implementan VLIW suele ser responsabilidad del software ordenar las instrucciones de manera que los datos estén disponibles al momento de ejecutar la instrucción.

Etapa de Asignación

- Asignación en procesadores out-of-order
  - Unified Issue Queue: una única queue con todas las instrucciones
  - Reservation Station: cada unidad de ejecución tiene su propia queue de instrucciones
  - Distributed Issue Queue: las unidades funcionales estan agrupadas y cada grupo tiene su propia issue queue
- Las instrucciones pueden obtener sus operandos antes de entrar a la queue (read before issue) o despues de entrar a la queue (read after issue)

Etapa de Asignación – read before issue

- La issue queue tiene información de la instrucción y los valores de los operandos de entrada
- El bloque Ctrl Info tiene información sobre la ejecución de la instrucción (tipo de ALU necesaria, tamaño de los datos, uso de operando inmediato,etc)
- Los bloques Src1Id y Src2Id tienen los identificadores de los registros que deben utilizarse (generados en la etapa de Renombrado)
- Los bloques V1 y V2 indican si el correspondiente SrcId tiene un dato valido o no.
- Los bloques SrcData1 y SrcData2 contienen los valores de los operandos
- Los bloques R1 y R2 indican si los datos en el correspondiente SrcData estan listos para ser usados.

Etapa de Asignación – read after issue

- La issue queue tiene información de la instrucción pero no de los valores de los operandos de entrada
- Es necesario tener un bloque para los valores inmediatos
- Se reduce el numero de ciclos entre la etapa de renombrado y la asignacion a la issue queue debido a que no es necesario leer datos.

Etapa de Asignación – operaciones con memoria

- Las dependencias de memoria no se pueden identificar en la etapa de renombrado, por lo que se resuelven en la etapa de asignación.
- Hay distintos mecanismos para gestionar las dependencias de memoria. Los mismos se clasifican en No Especulativos y Especulativos
- Los mecanismo No Especulativos no permiten una operación en memoria hasta no asegurar que no hay dependencias con otra operación en memoria previa.
- Los mecanismos Especulativos tratan de predecir si va a haber una dependencia. En caso de dependencia, hay distintos algoritmos para gestionarla.

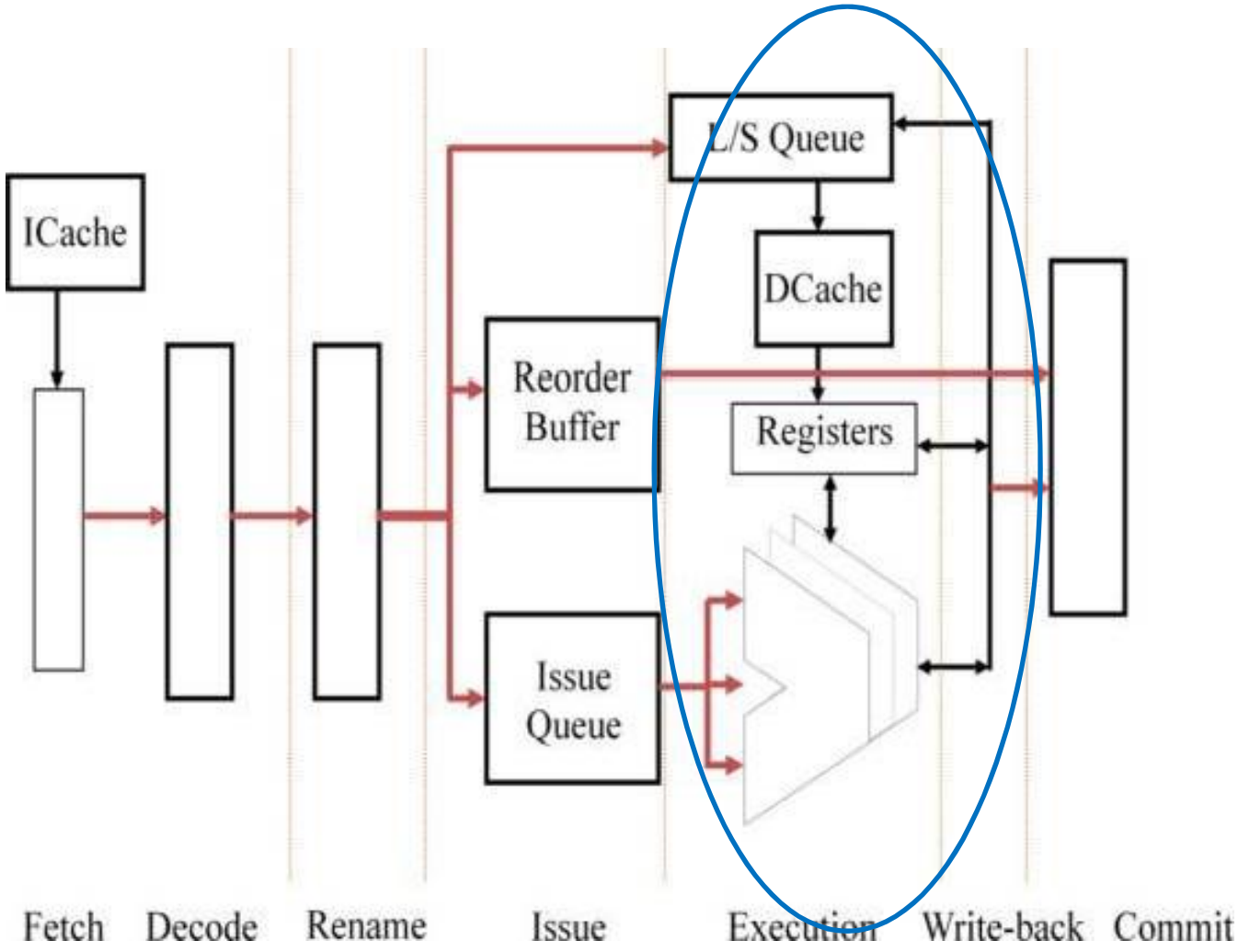
Etapa de Asignación – operaciones con memoria no especulativas

- Orden Total: todos los accesos a memoria se realizan en el orden que aparecen en el programa
- Orden Parcial: Todas las escrituras a memoria se realizan en orden pero las lecturas se realizan out-of-order una vez realizadas todas las escrituras previas.
- Orden de Lectura y Escritura: la ejecución de lecturas y escrituras es out-of-order, pero todas las lecturas son in-order entre si y todas las escrituras son in-order entre si.

Etapa de Asignación – operaciones con memoria especulativas

- Orden de Almacenamiento: Las escrituras se realizan in-order pero las lecturas se realizan out-of-order (Intel Core)
- Se realizan especulativamente las lecturas que se predice que no son dependientes de las instrucciones actualmente en proceso.
- La predicción puede fallar, por lo que requiere hardware especial para recuperar el procesador al estado previo y continuar la ejecución.

Etapa de Ejecución (execution)

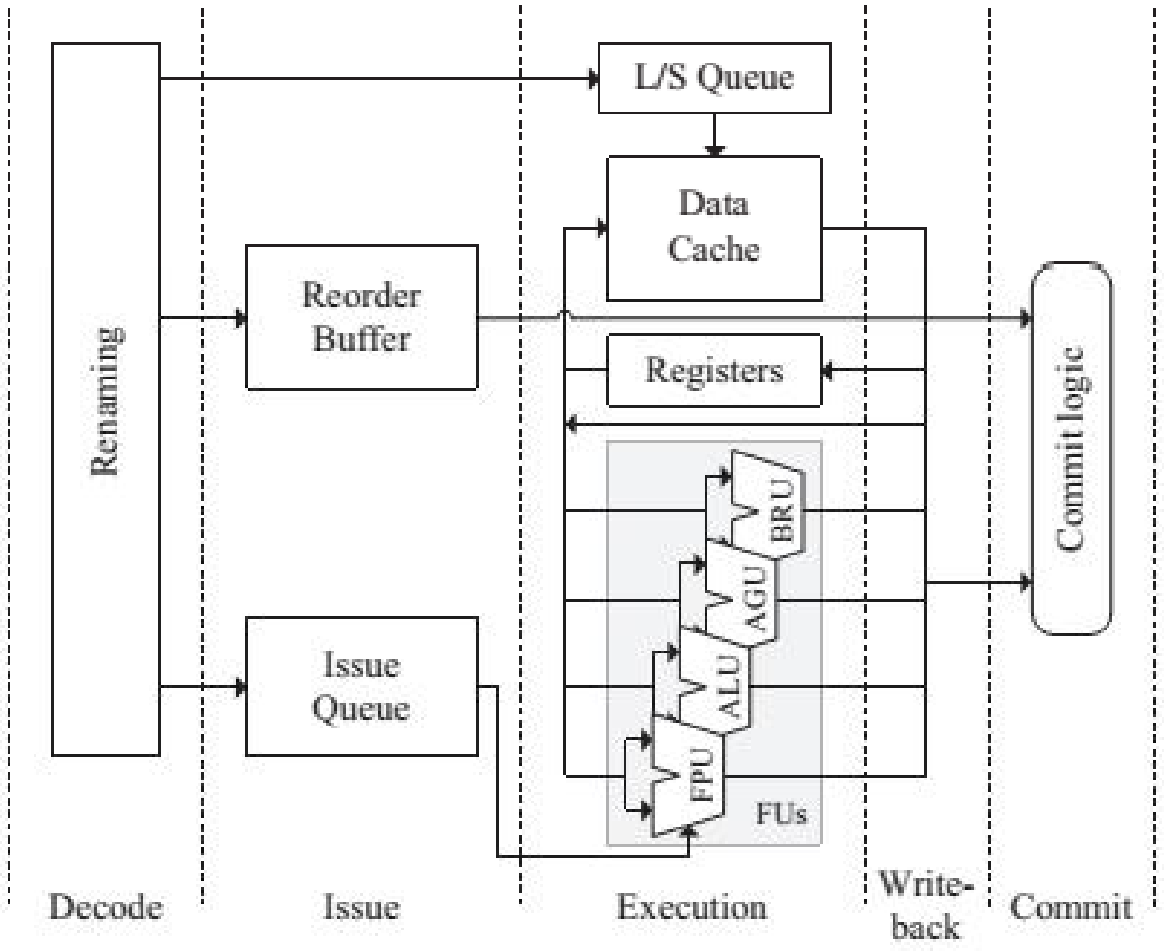


ETAPA DE EJECUCIÓN

- En esta etapa se procesan los operandos de la instrucción en la unidad funcional correspondiente.
- Los distintos tipos de instrucciones (aritméticas, memoria, bifurcación, etc) tienen distinta complejidad, y por ende, distinta latencia.
- Las unidades funcionales se dividen en:
  - FPU: Operaciones de punto flotante
  - ALU: Operaciones aritmético-lógicas
  - AGU: Address Generation Unit, calculo de direcciones
  - BRU: Branch Unit, procesa las bifurcaciones.
  - SIMD: Single Instruction, Multiple Data, misma operación sobre varios operandos

Etapa de ejecución

- Diagrama

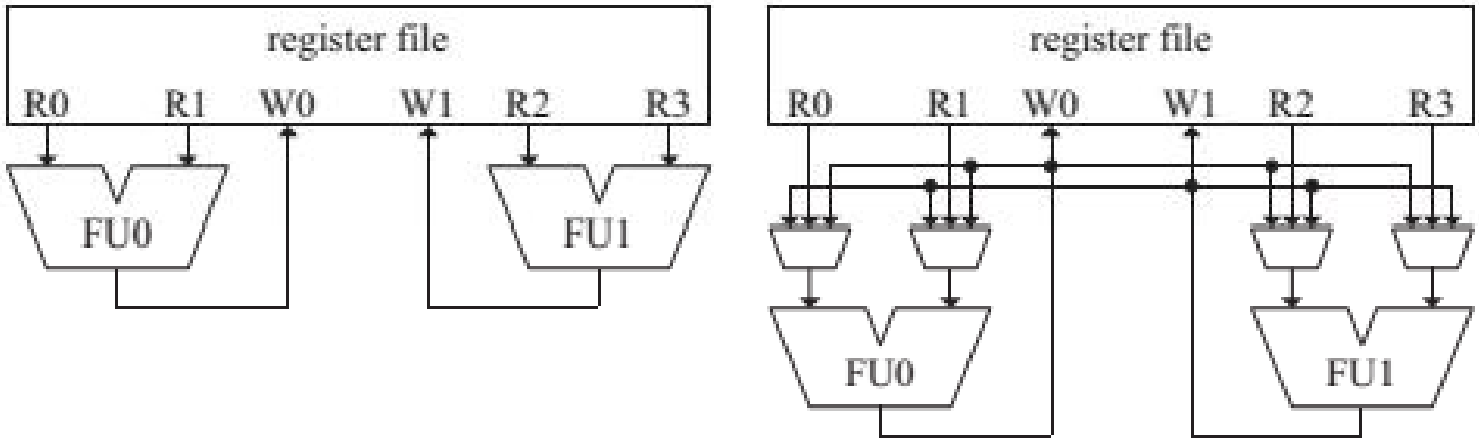


ETAPA DE EJECUCIÓN

- Bypass Network: distribuyen los resultados entre las distintas unidades funcionales de acuerdo a su dependencia, al Data Cache y Architectural Register File.
- Los resultados se consideran Especulativos hasta la finalización de la etapa de Remisión, donde se copian al Architectural Register File o se escriben en el Data Cache.
- El bypass es distinto según si el procesador es out-of-order o in-order.

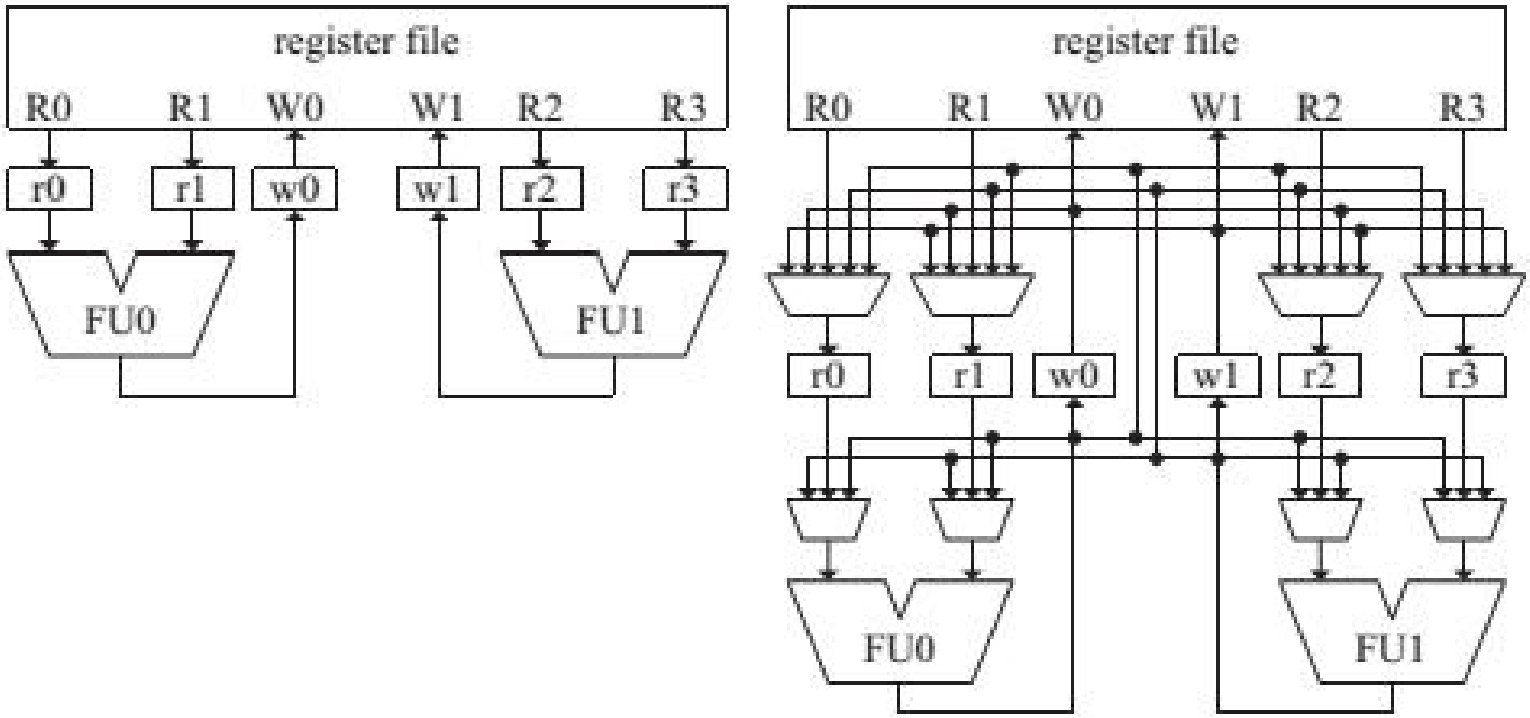
ETAPA DE EJECUCIÓN – BYPASS EN OUT-OF-ORDER (SIMPLE)

- Diagrama (izq. sin bypass, der. con bypass)



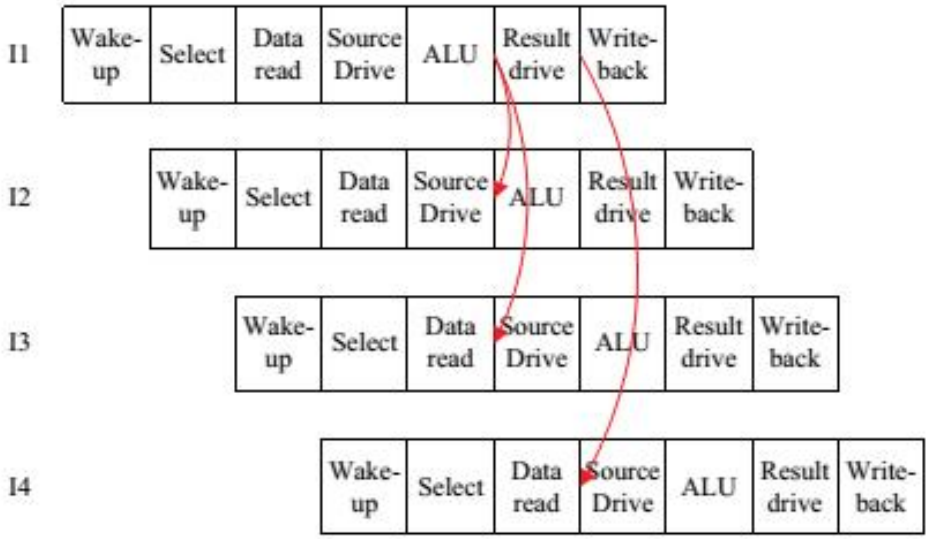
Etapa de ejecución – Bypass en out-of-order (Multilevel)

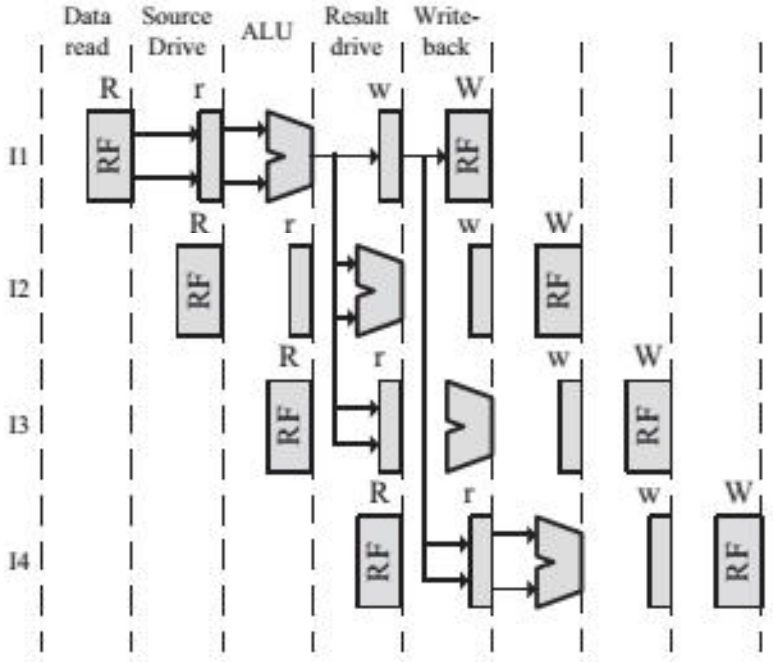
- Diagrama (izq. sin bypass, der. con bypass)



Etapa de ejecución – Bypass en out-of-order (Multilevel)

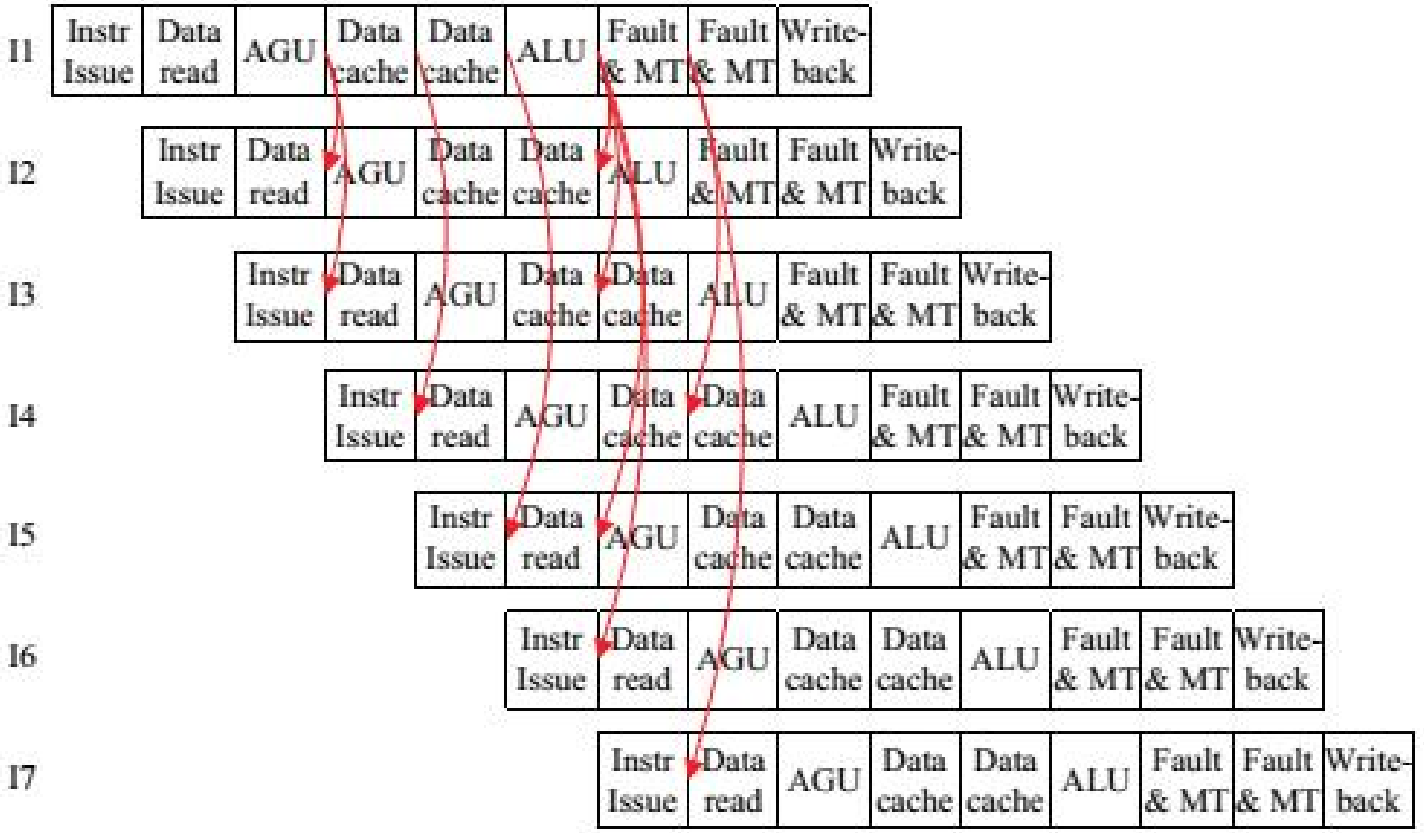
- Diagramas Temporales





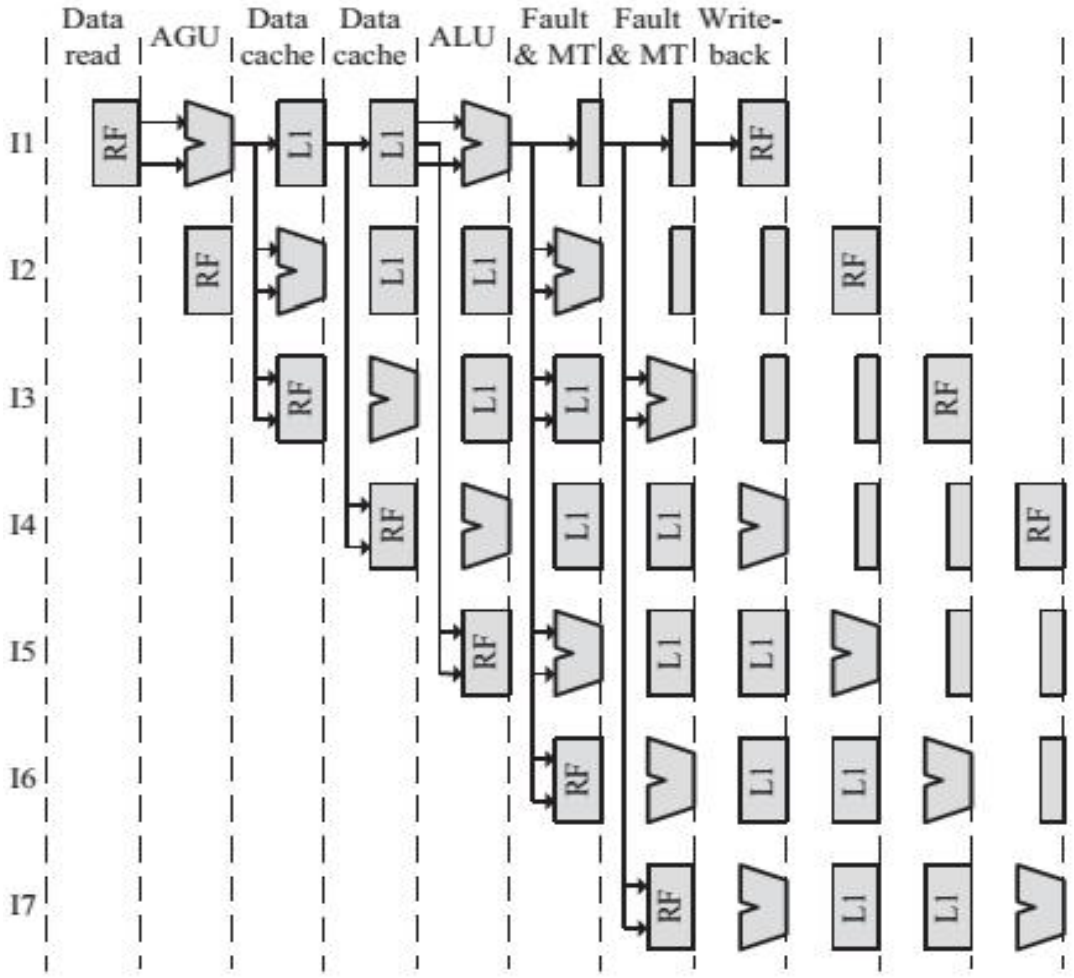
Etapa de ejecución – Bypass en in-order (Multilevel)

- Diagramas Temporales (Intel Atom)

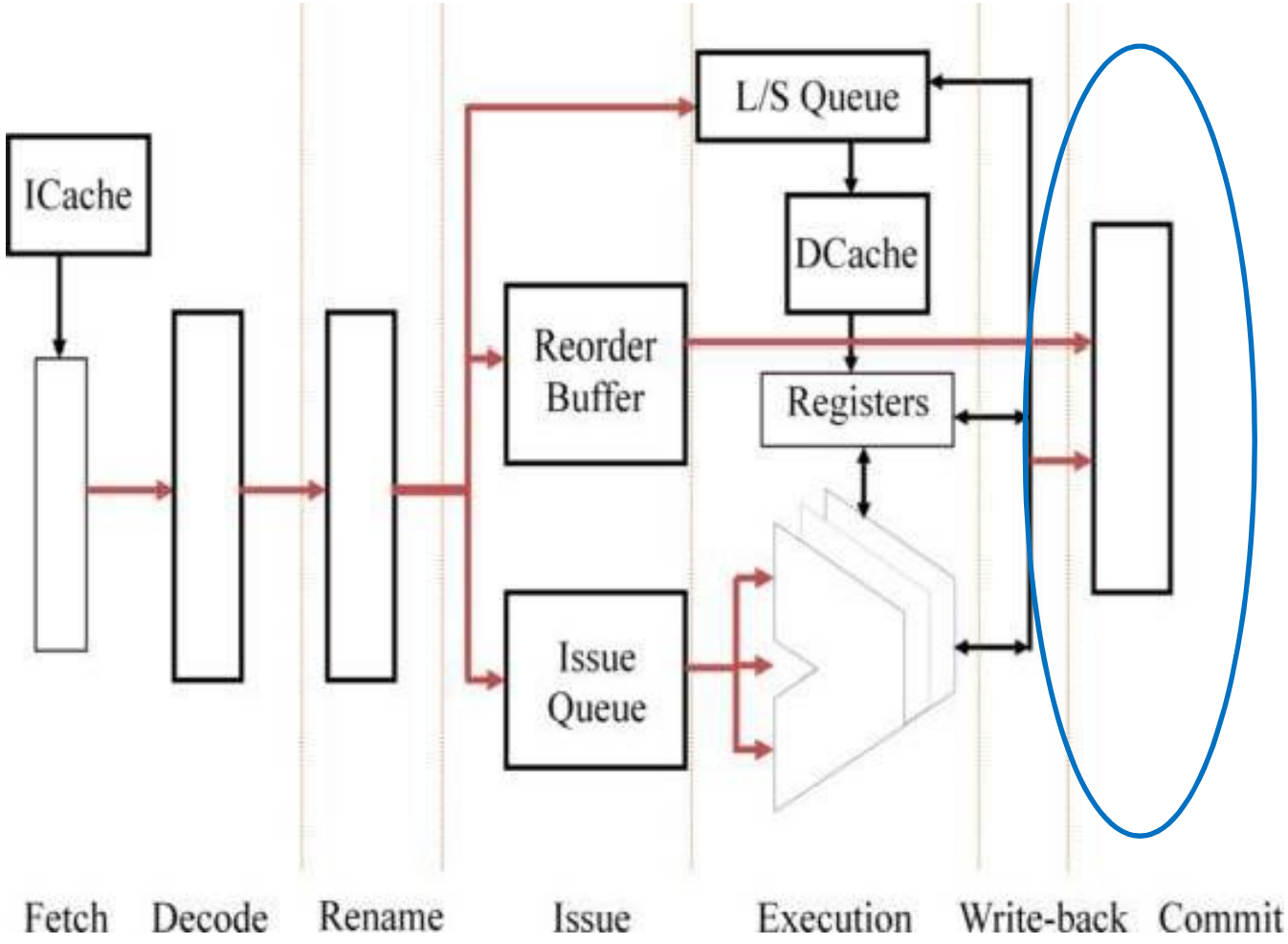


Etapa de ejecución – Bypass en in-order (Multilevel)

- Diagramas Temporales (Intel Atom)



Etapa de Remisión (commit)

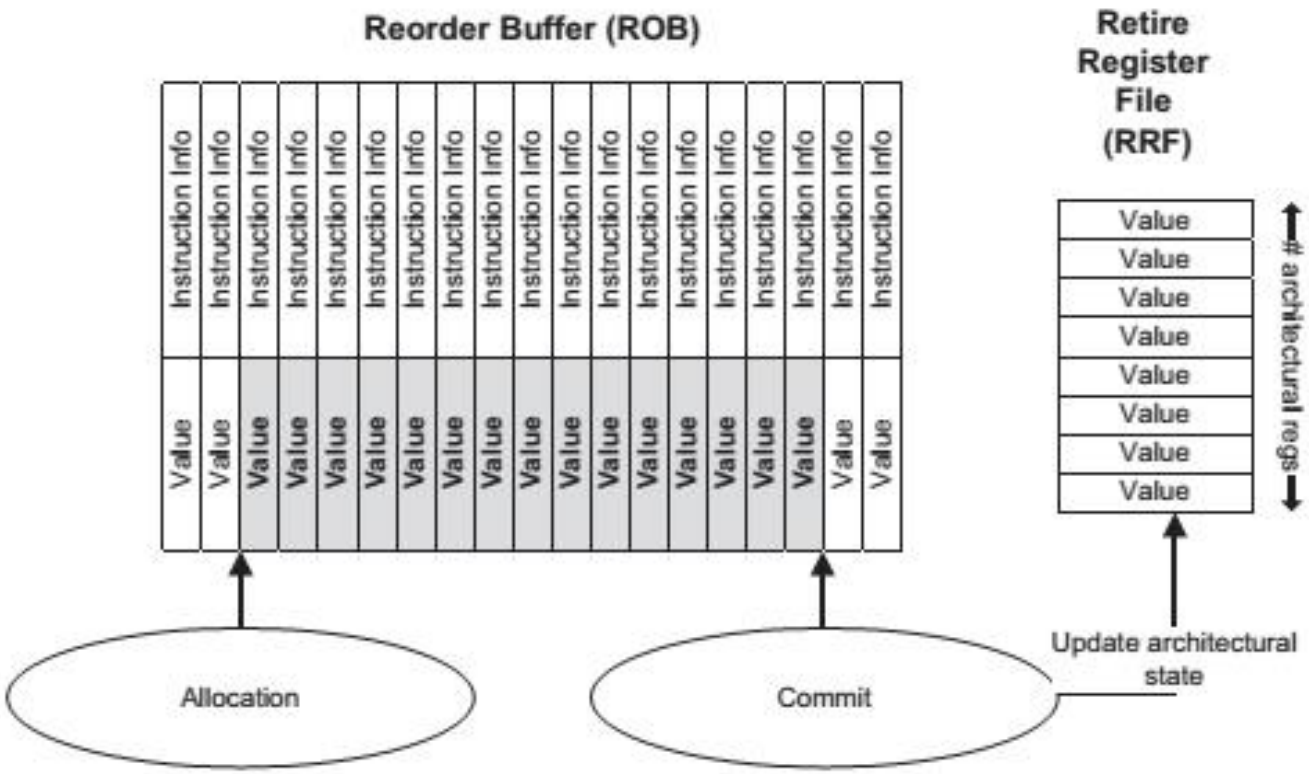


Etapa de Remisión

- Esta etapa se implementa principalmente en procesadores out-of-order para emular la ejecución in-order.
- Los resultados pasan a los registros en el orden que esta en el programa
- Los recursos utilizados por la instrucción son liberados (entradas en el ROB, buffers y tablas intermedias, etc)

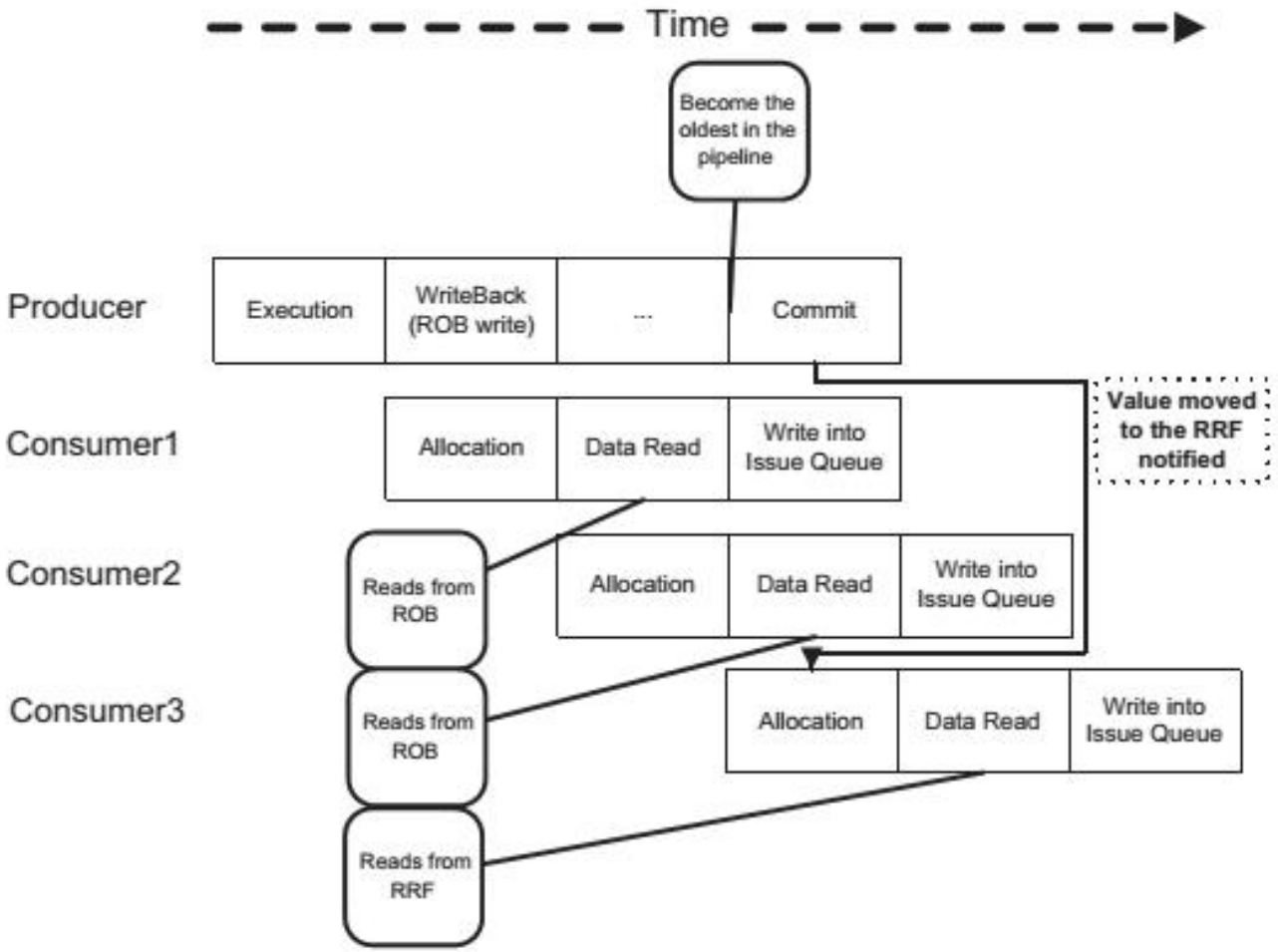
Etapa de Remisión

- Transferencia del ROB al ARF (tambien se denomina Retire Register File – RRF)



Etapa de Remisión

- Evolución temporal



Etapa de Remisión

- En esta etapa tambien se realiza la recuperación cuando el estado especulativo no es correcto.
- Esto se puede deber a una mala predicción de bifurcación o porque una instrucción genera una excepción.

Etapa de Remisión

- Recuperación de una mala predicción de bifurcación
  - Se divide en la recuperación del front-end y del back-end
  - Front-end:
    - Limpieza de cache y buffers de instrucciones que corresponden a la bifurcación errónea hasta la etapa de Renombrado
    - Reinicializar el predictor de bifurcación
    - Actualizar el PC para tomar instrucciones de la bifurcación correcta.
  - Back-end:
    - Limpieza de cache y buffers de instrucciones que corresponden a la bifurcación errónea desde la etapa de Renombrado en adelante.
    - Reinicializar el predictor de bifurcación
    - Actualizar el PC para tomar instrucciones de la bifurcación correcta.

Etapa de Remisión

- Recuperación de una Excepción
  - Se hace en esta etapa para asegurar que la Excepción no es Especulativa
  - Hay que asegurar que el estado del procesador corresponda a la ejecución in-order de las instrucciones hasta la última antes de la Excepción.
  - Hay que vaciar buffers y pipeline de instrucciones en proceso porque la Excepción debe procesarse antes de continuar la ejecución del programa
  - El PC se redirecciona para obtener las instrucciones correspondientes al handler de la excepción

BIBLIOGRAFIA

- "Computer Architecture – A Quantitative Approach (5th Ed)" Hennessy & Patterson, Edit. MK, 2012
  - Texto clásico de arquitectura de computadoras. Orientado al hardware.
- "Computer Organization and Design – The Hardware / Software Interface (5th Ed)" Hennessy & Patterson, Edit. MK, 2014
  - Texto orientado a la integración software-hardware a nivel sistema
- "Digital Design and Computer Architecture (2nd Ed)" Harris & Harris, Edit. MK, 2013
  - Texto básico, empieza desde lo elemental de técnicas digitales, sirve como intruducción al tema.
- "Processor Microarchitecture – An Implementation Perspective" Gonzalez, Latorre & Magklis, Edit. M&C, 2011
  - Texto breve y conciso, sirve como resumen y panorama global