

Arquitectura de Computadores

Grado de Informática

Tema I: Análisis y Diseño de Procesadores Superescalares

Carlos Molina Clemente

Universitat Rovira i Virgili – Tarragona, Catalunya, Spain

carlos.molina@urv.net



Tema 1: Análisis y Diseño de Procesadores Superescalares (AC, Grado de Informática, ETSE, URV)

Temario

- I. Análisis y Diseño de Procesadores Superescalares
- II. Evaluación del Rendimiento, Consumo y Coste
- III. Análisis de Procesadores Paralelos
- IV. Optimización de Programas

Índice Tema I

1. Introducción

2. Riesgos: Estructurales, Datos y Control

3. Camino de Datos del Procesador Superescalar

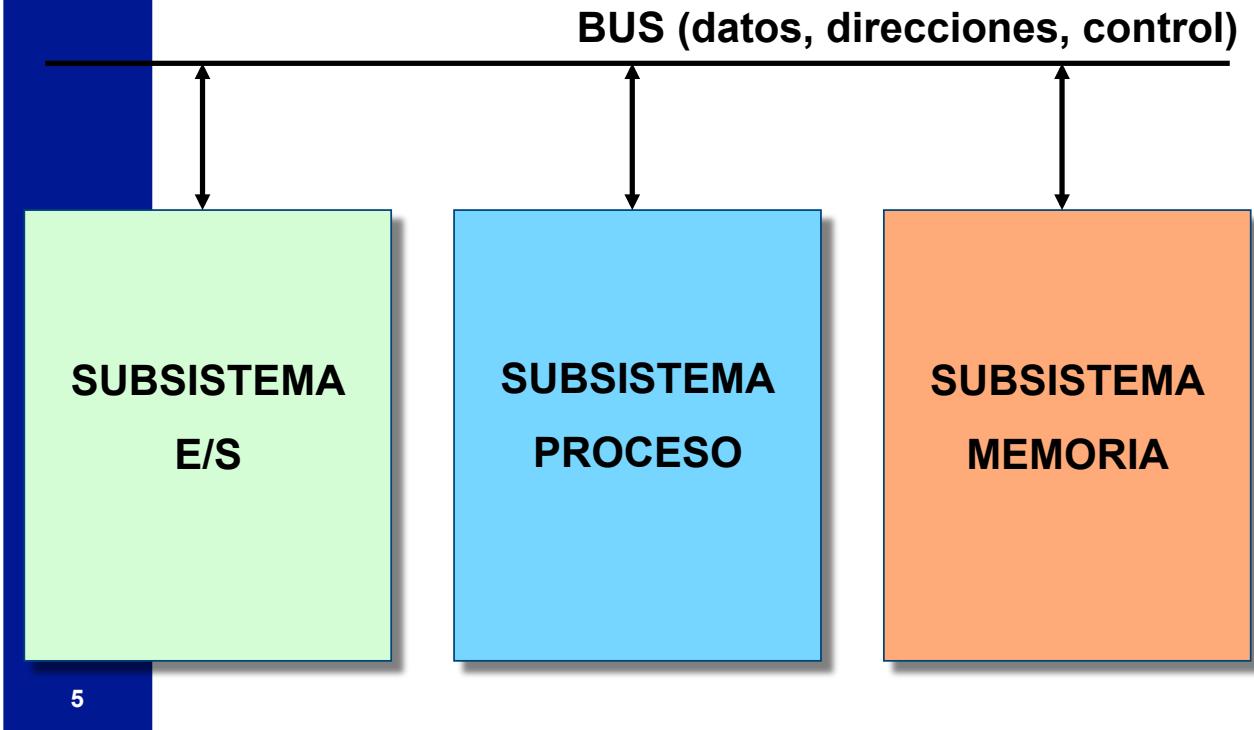
4. Técnicas Microarquitectónicas Avanzadas

5. Procesadores VLIW

Introducción

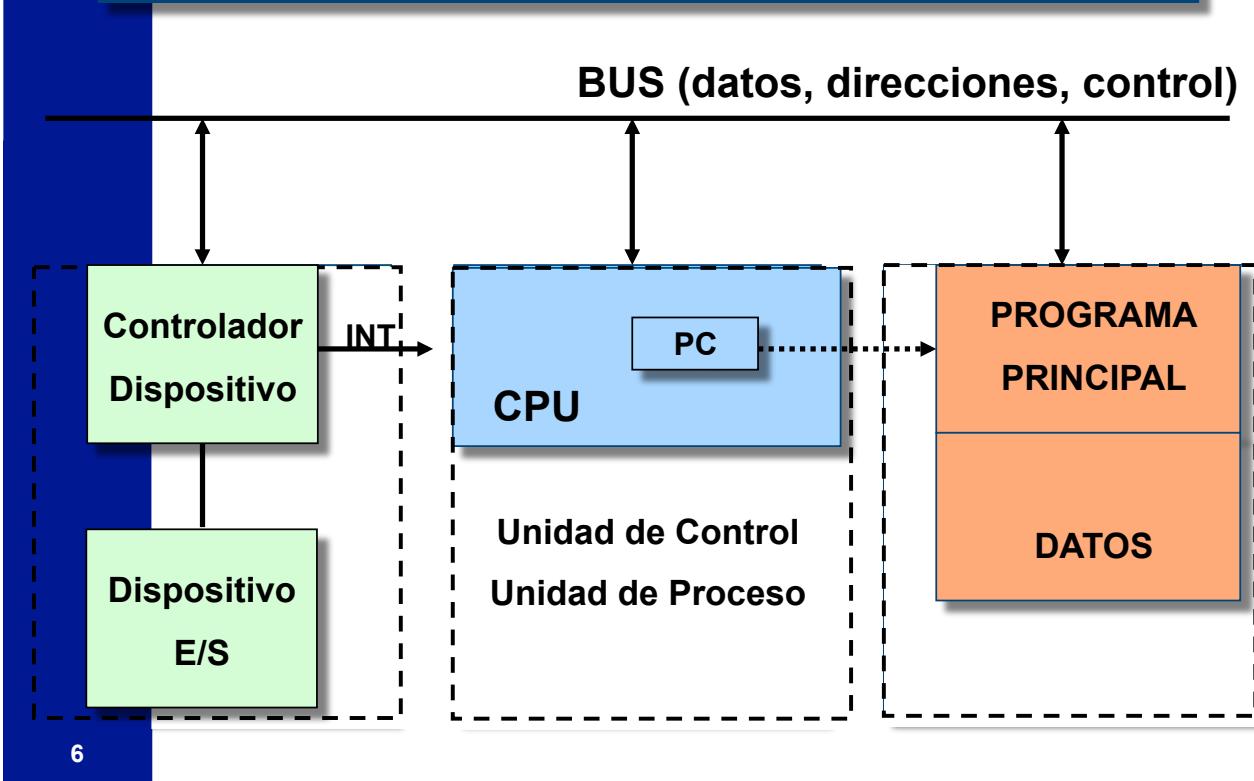
- **Conceptos Básicos**
- **Modelo de Ejecución en Etapas**
- **Pipeline del MIPS**
- **Historia Superescalares**

Arquitectura Von Neumann



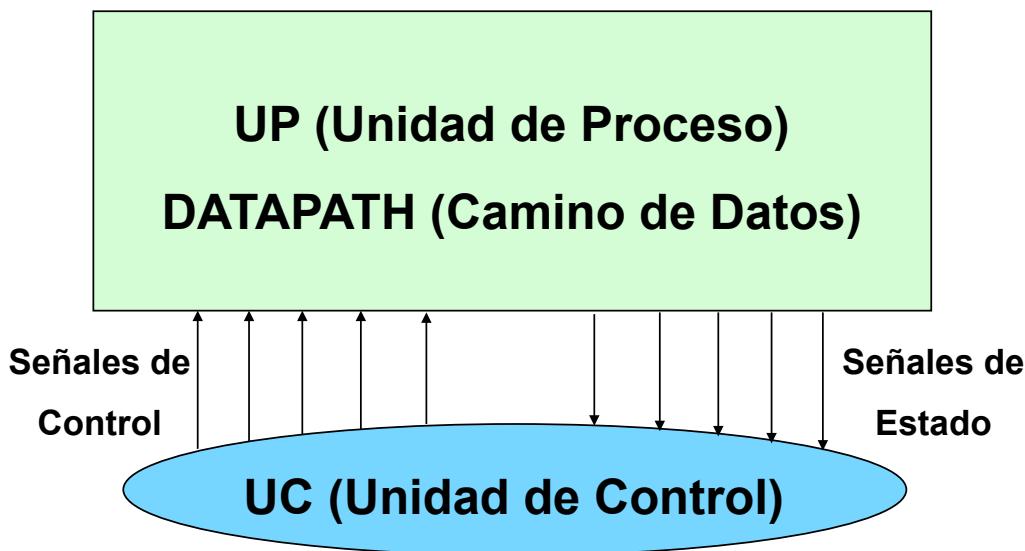
5

Arquitectura Von Neumann



6

CPU (Control Process Unit)



OBJETIVO: Conocer las características y estructuras básicas de un Procesador Superescalar

7

Rendimiento (Factores Influencia)

$$T_{CPU} = I_c * CPI * \tau$$

$$CPI = p + (m * k)$$

- I_c : número instrucciones
- CPI: ciclos x instrucción
- τ : periodo = 1/frecuencia = tiempo de ciclo
- p : ciclos decodificación/ejecución instrucción
- m : porcentaje de referencias a memoria
- k : ciclos de operación de acceso a memoria

	I_c	CPI			τ
		p	m	k	
Juego de instrucciones	X	X			
Tecnología del compilador	X	X	X		
Implementación del procesador		X		X	X
Escala de integración		X		X	X

8

CPI y Tiempo de Ciclo (P.E.Monociclo)

■ Relacionados y dependientes de la implementación

■ Tipos de implementación

1. Procesador Escalar Monociclo
2. Procesador Escalar Multiciclo
3. Procesador Segmentado
4. Procesador Superescalar



1. Procesador Escalar Monociclo

- Se opera con una sola instrucción en cada ciclo
- Cada instrucción tarda 1 ciclo

$$\text{CPI} = 1 \quad T_{\text{ciclo}} \uparrow$$

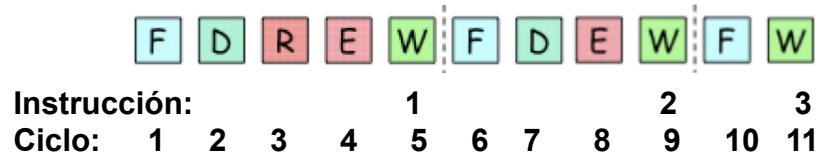


9

CPI y Tiempo de Ciclo (P.E.Multiciclo)

2. Procesador Escalar Multiciclo

- Se opera con una sola instrucción en cada ciclo
- Cada instrucción se divide en varios pasos
- Cada paso tarda 1 ciclo



10



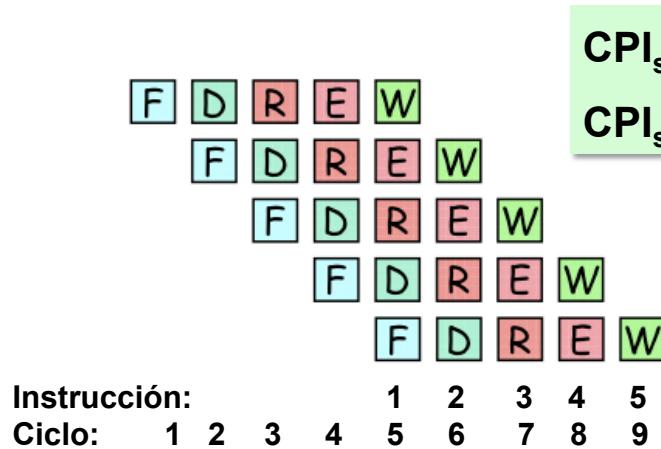
$$\text{CPI} > 1 \quad T_{\text{ciclo}} \downarrow$$

CPI y Tiempo de Ciclo (P.Segmentado)

3. Procesador Segmentado

- Cada instrucción se divide en varios pasos
- Cada paso tarda 1 ciclo
- Se inicia la ejecución de una instrucción cada ciclo

11



$\text{CPI}_{\text{segmentado}} < \text{CPI}_{\text{multiciclo}}$
 $\text{CPI}_{\text{segmentado}} \approx 1 \quad T_{\text{ciclo}} \downarrow$

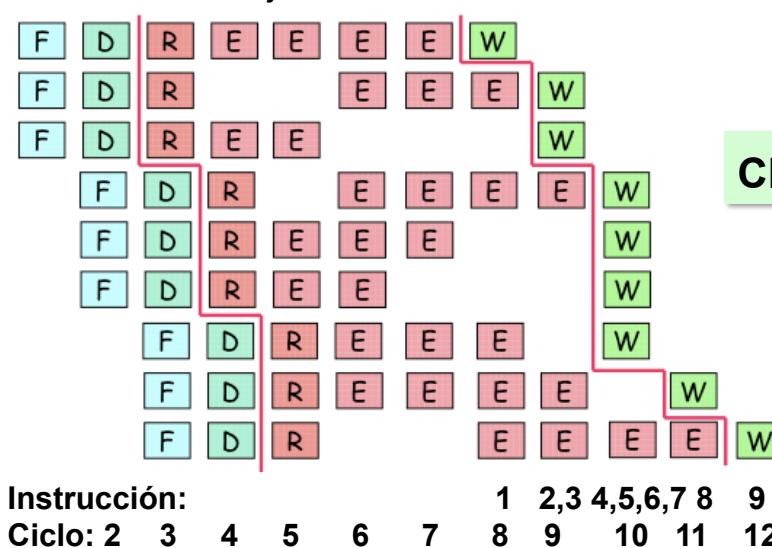


CPI y Tiempo de Ciclo (P.Superescalar)

4. Procesador Superscalar

- Cada instrucción se divide en varios pasos
- Cada paso tarda 1 ciclo
- Se inicia la ejecución de **VARIAS** instrucciones cada ciclo

12



$\text{CPI} < 1 \quad T_{\text{ciclo}} \downarrow$



Taxonomía de Flynn (1972)

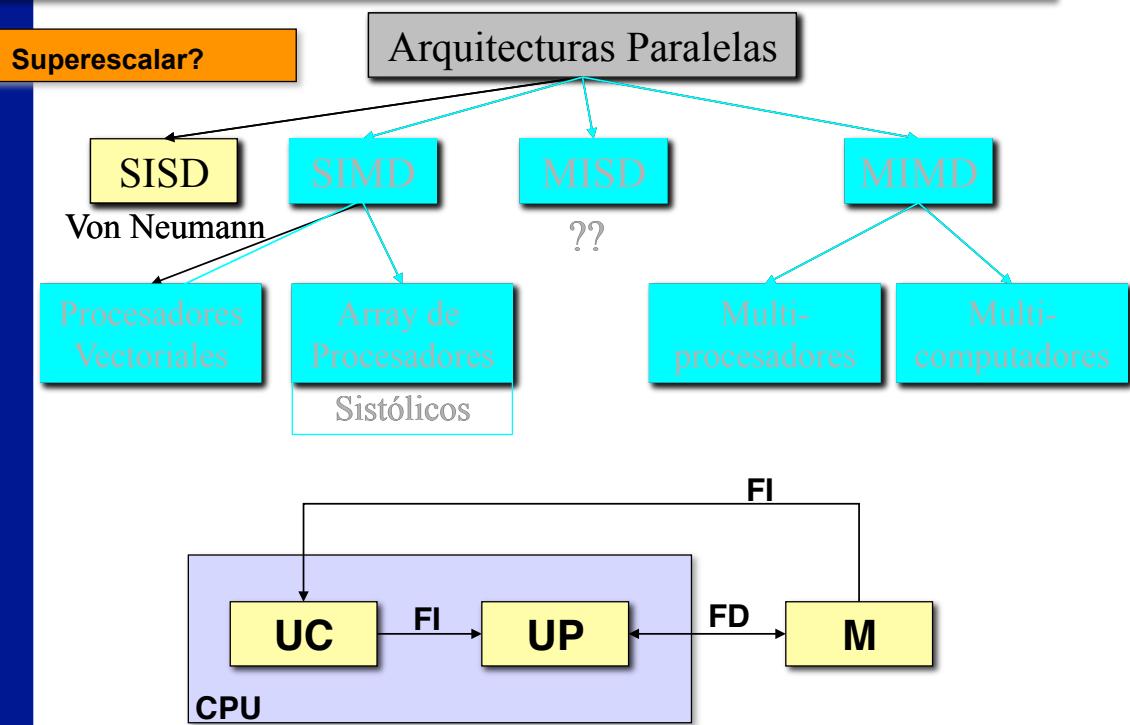
- Combina Flujo de Datos y Flujo de Instrucciones con (Single) Único y Múltiple, dando 4 combinaciones:

		Flujo Datos	
		Único	Múltiple
Flujo Intrucc.	Único	SISD	SIMD
	Múltiple	MISD	MIMD

SISD	Una Instrucción un Dato	Von Neumann
SIMD	Una Instrucción muchos Datos	Vectoriales/Sistólicos
MISD	Muchas Instrucciones un Dato	Teórico
MIMD	Muchas Instrucciones muchos Datos	Multiprocesadores / Multicomputadores

13

Taxonomía Flynn (SISD)



14

Taxonomía Flynn (MIMD)

Superescalar?

Arquitecturas Paralelas

SISD

Von Neumann

SIMD

Procesadores
Vectoriales

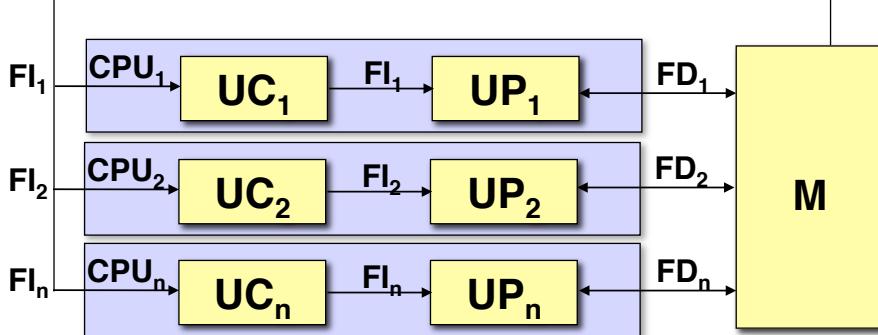
MISD

??

MIMD

Multi-
computadoresArray de
ProcesadoresMulti-
procesadores

Sistólicos



15

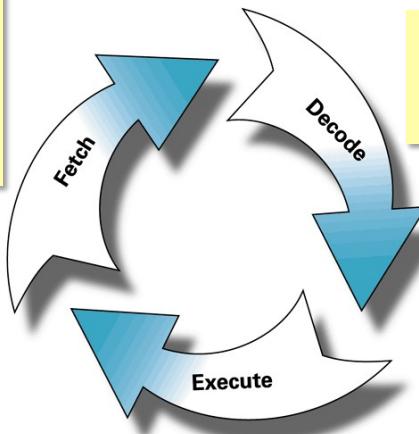
Introducción

- Conceptos Básicos
- Modelo de Ejecución en Etapas
- Pipeline del MIPS
- Historia Superescalares

16

Etapas de una Instrucción

1. Recuperar la siguiente instrucción desde memoria (apuntada por el *program counter*) y luego incrementar el *program counter*.



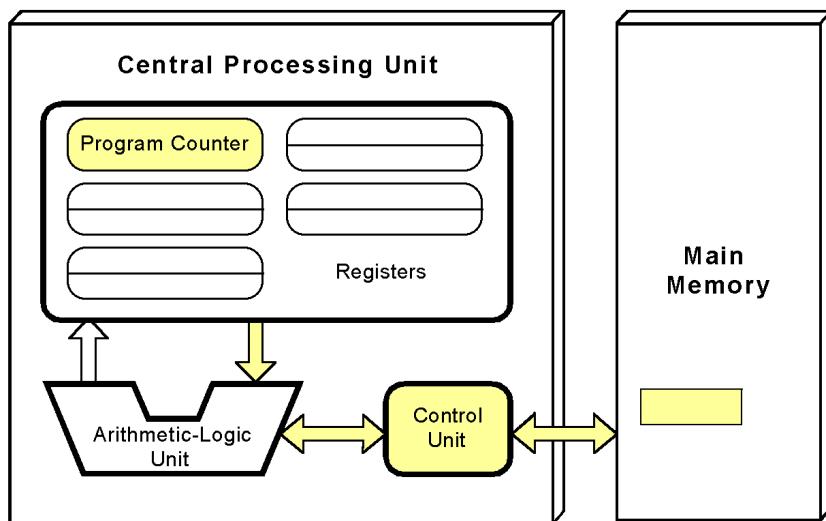
2. Decodificar el patrón de bits en el registro de instrucción IR

3. Ejecutar la instrucción indicada en el registro de instrucción IR

17

Fetch

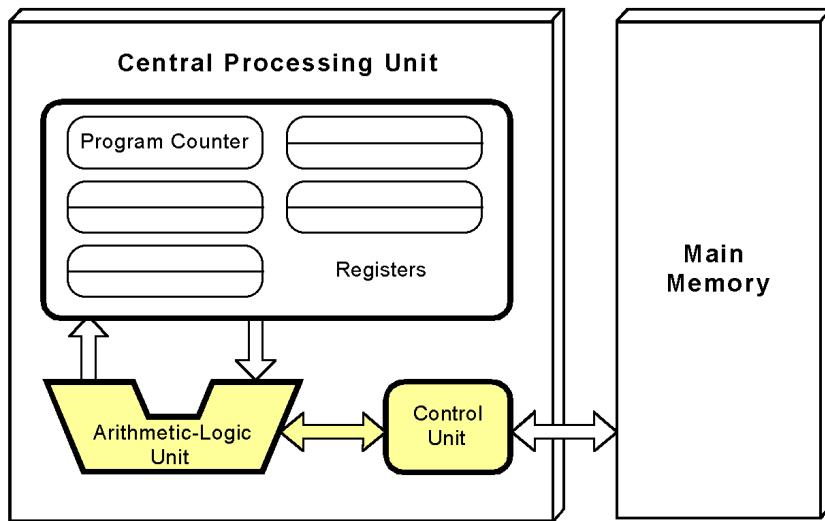
La unidad de control obtiene la próxima instrucción de memoria usando el “contador de programa” (PC) que dice en qué dirección se encuentra.



18

Decode

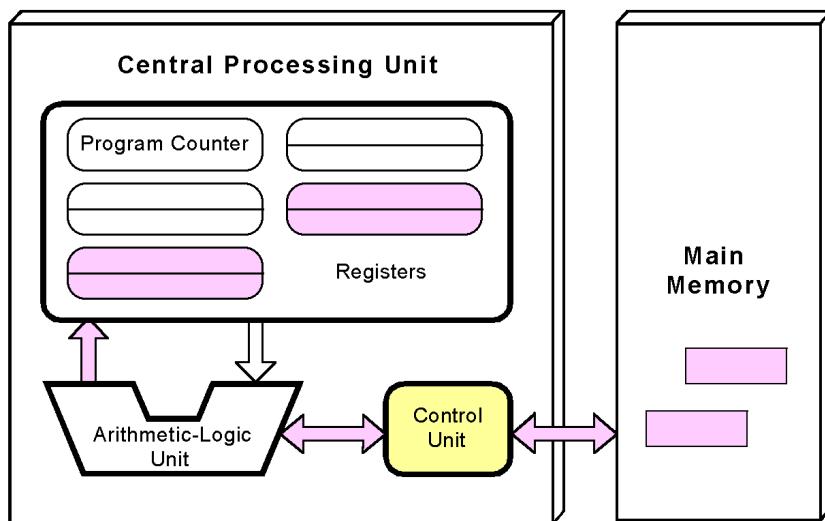
La instrucción se decodifica a un lenguaje que entiende la ALU (unidad aritmética lógica).



19

Decode

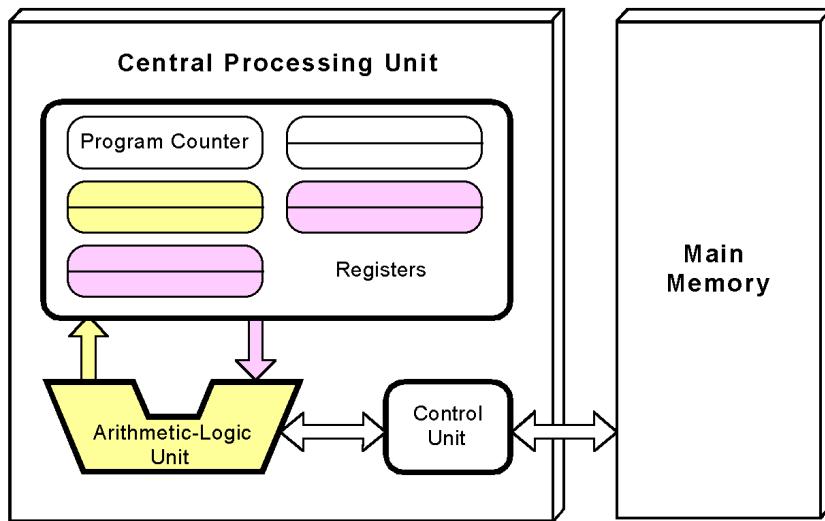
Las instrucciones leen sus operandos de registros que se han generado con anterioridad de otros cálculos o se traen de memoria



20

Execute

La ALU ejecuta la instrucción y coloca los resultados en registros o en memoria.



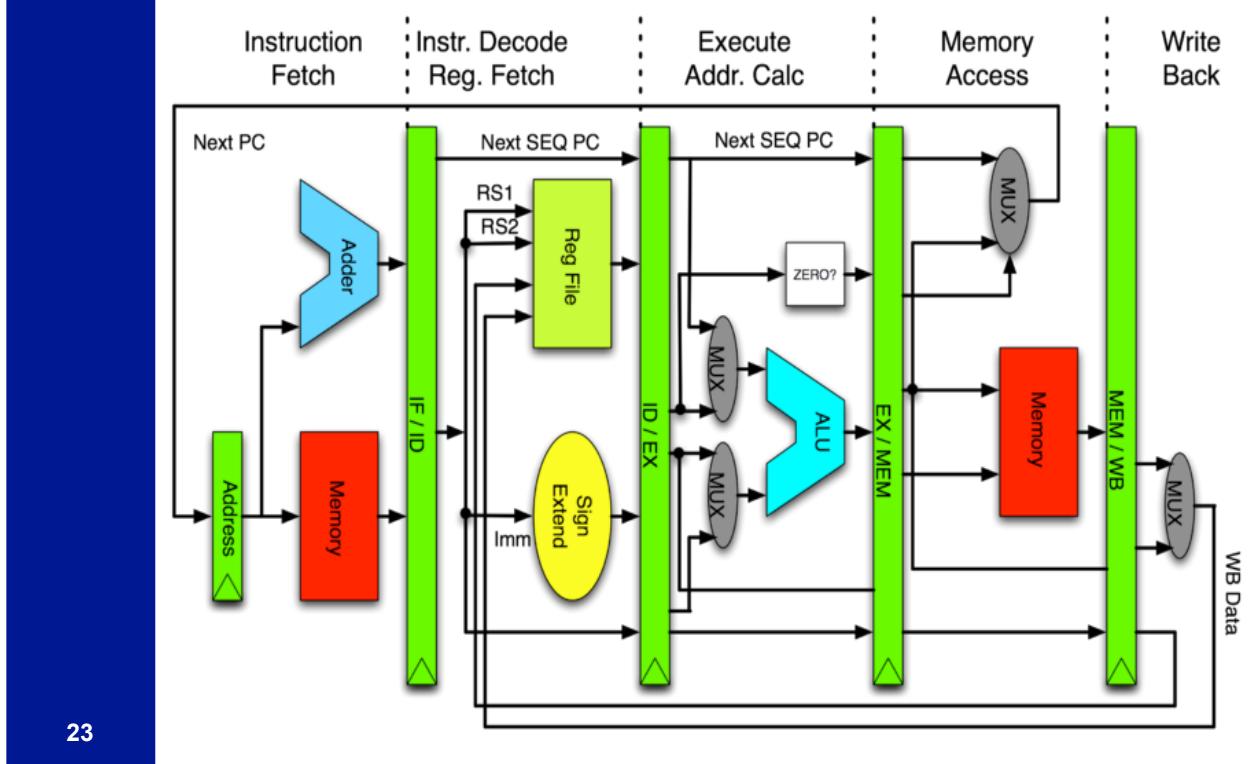
21

Introducción

- Conceptos Básicos
- Modelo de Ejecución en Etapas
- Pipeline del MIPS
- Historia Superescalares

22

Ejemplo: MIPS Segmentado



23

Pipeline MIPS (Etapas)

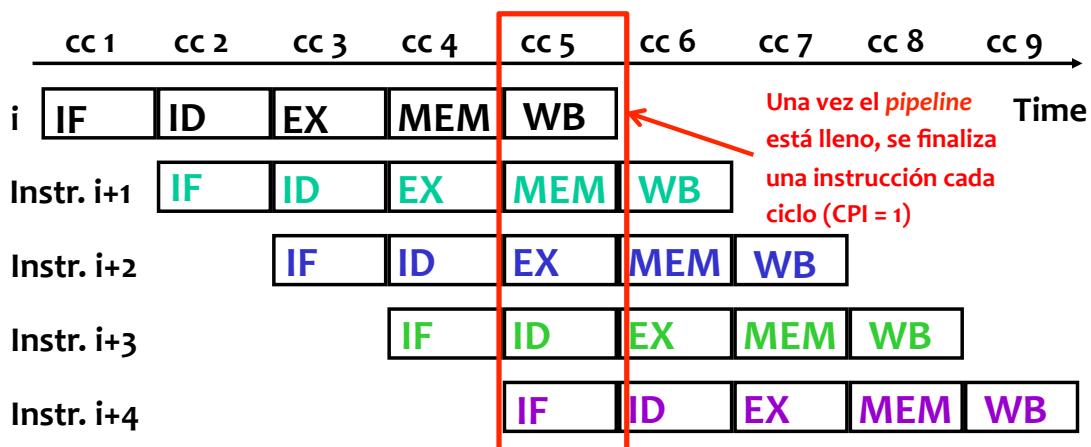
1. **IF: (Instruction Fetch)** Búsqueda de la instrucción, actualización del PC
2. **ID: (Instruction Decode)** Decodificación de la instrucción, lectura del banco de registros
3. **EX: (Execution)**
 - Instrucción de acceso a memoria: Cálculo de la dirección
 - Instrucción aritmético-lógica: Cálculo de la operación
4. **MEM: (Memory)**
 - **Load:** Lectura de la memoria de datos
 - **Store:** Escritura de la memoria de datos
5. **WB: (Write Back)** Escritura de resultado en banco registros

24

Pipeline MIPS (CPI)

- Cada etapa del *pipeline* se completa en 1 ciclo de reloj
- Cada instrucción usa mismo número de etapas para ejecutarse, aunque algunas estén inactivas
- Mejora throughput pero latencia de Inst. NO se reduce

25



Pipeline MIPS (Tiempo de Ciclo)

- Comparamos ciclo de reloj monociclo vs segmentado
- Asumimos latencias de:
 - Unidades de memoria: 200 ps
 - ALU y sumadores: 200 ps
 - Banco de registros (R/W): 100 ps

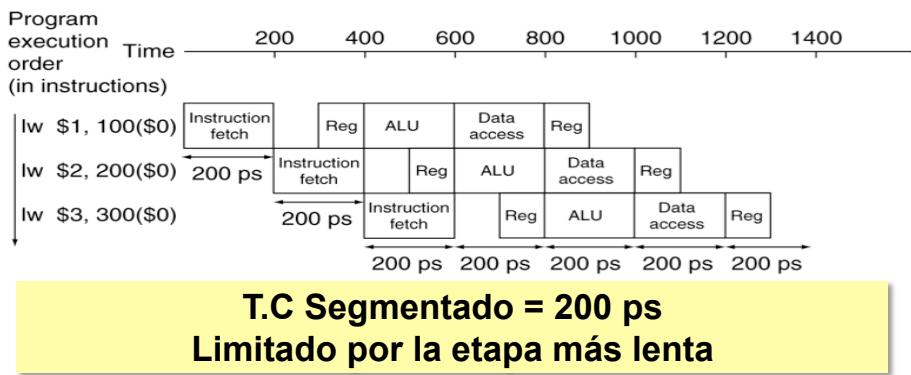
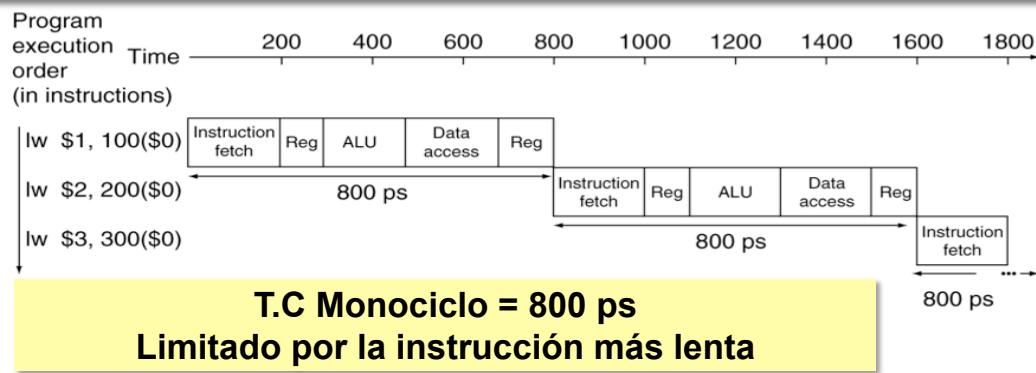
resto componentes
(cables, etc) se consideran
latencias despreciables

Tipo de instrucción	Memoria de instrucciones	Lectura banco de registros	Operación ALU	Memoria de datos	Escritura banco de registros	Total
Tipo R	200	100	200	0	100	600 ps
Load word	200	100	200	200	100	800 ps
Store word	200	100	200	200		700 ps
Branch	200	100	200			500 ps

T. Ciclo de reloj para la versión monociclo: 800 ps
 T. Ciclo de reloj de la versión segmentada: 200 ps

26

Pipeline MIPS (Tiempo de Ciclo)



27

Pipeline MIPS (Speedup)

- Si todas las etapas están balanceadas, es decir, todas tardan el mismo tiempo:

$$\text{Tiempo entre instrucciones}_{\text{segmentado}} = \frac{\text{Tiempo entre instrucciones}_{\text{no segmentado}}}{\text{Número de etapas del pipeline}}$$

- En condiciones ideales (etapas balanceadas) y con un número alto de instrucciones el speedup es igual al número de etapas

- Segmentación eleva rendimiento:

- incrementando tasa de ejecución de instrucciones (**throughput**),
- sin reducir tiempo de ejecución (**latencia**) de instrucción individual

28

Pipeline MIPS (ISA)

- ISA MIPS diseñado para ejecución segmentada
- Todas las instrucciones son de la misma longitud
 - Facilita búsqueda de instrucciones (IF) y decodificación (DE)
- Pocos formatos de instrucción
 - Campos registros fuente ocupan mismo lugar en cada formato
 - Permite en 2^a etapa leer banco y determinar tipo de instrucción
- Operандos de memoria sólo en *load* y *store*
 - Etapa EX calcula @ y acceso se realiza en siguiente etapa (MEM)
- Cada instrucción escribe a lo sumo un resultado y en las últimas etapas (MEM, WB)
- Alineamiento de memoria
 - Los accesos a memoria se realizan en un solo ciclo de reloj

29

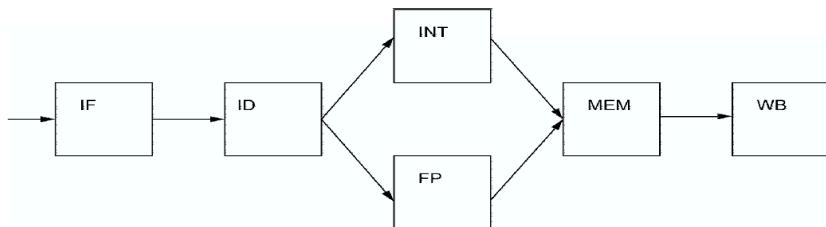
Introducción

- Conceptos Básicos
- Modelo de Ejecución en Etapas
- Pipeline del MIPS
- Historia Superescalares

30

Historia

- El primer paso hacia una arquitectura superescalar fue agregar otra unidad de ejecución al pipeline.



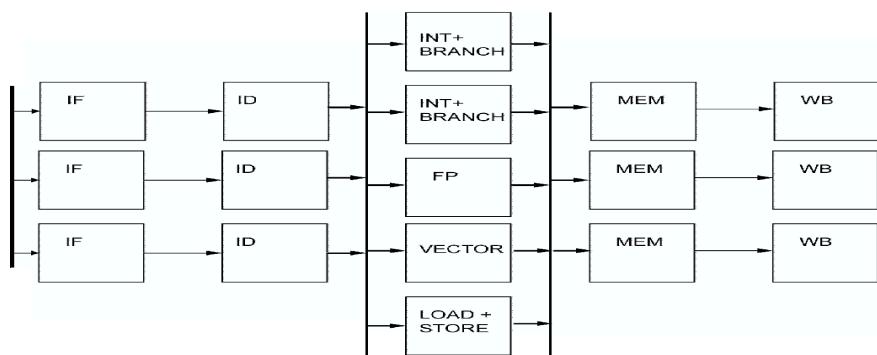
- Supercomputadoras como Control Data 6600 y 7600, Cray-1 y algunas mainframes de IBM (años 60)



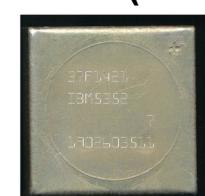
31

Historia

- El siguiente paso era duplicar el pipeline



- Procesador IBM Power1 en la RS/6000 (1990)



32

Superescalares Comerciales

Microprocessor	Year	Clock Rate	Pipeline Stages	Issue width	Out-of-order/Speculation	Cores	Power
i486	1989	25MHz	5	1	No	1	5W
Pentium	1993	66MHz	5	2	No	1	10W
Pentium Pro	1997	200MHz	10	3	Yes	1	29W
P4 Willamette	2001	2000MHz	22	3	Yes	1	75W
P4 Prescott	2004	3600MHz	31	3	Yes	1	103W
Core	2006	2930MHz	14	4	Yes	2	75W
UltraSparc III	2003	1950MHz	14	4	No	1	90W
UltraSparc T1	2005	1200MHz	6	1	No	8	70W

Issue Width también se conoce como GRADO del procesador

33

Superescalares (Año 2010)

■ Intel Core i7 Nehalem

- cores: 6
- pipeline stages : 24
- issue: 4



■ IBM POWER 7

- cores: 8
- pipeline stages: 20
- issue: 6



■ SUN SPARC T3 (Niagara 3)

- cores: 8,16
- pipeline stages: 12
- issue: 2



Procesador Superescalar: Múltiple Issue Dinámico

34

Índice Tema I

1. Introducción

2. Riesgos: Estructurales, Datos y Control

3. Camino de Datos del Procesador Superescalar

4. Técnicas Microarquitectónicas Avanzadas

5. Procesadores VLIW

Riesgos (Hazards)

- El principio de la segmentación es simple pero su implementación introduce RIESGOS o HAZARDS
 - Problemas o situaciones en las cuales la próxima instrucción no puede ejecutarse en el siguiente ciclo de reloj
- Pueden obligar a detener (*stall*) el *pipeline*, evitando que se logre el *speedup* ideal
 - Los riesgos se pueden resolver esperando, pero será el controlador del *pipeline* quien los detecte y resuelva
- Tipos:
 - Riesgos estructurales (*Structural hazards*)
 - Riesgos de datos (*Data hazards*)
 - Riesgos de control (*Control hazards*)

Riesgos

- **Riesgos Estructurales**
- **Riesgos de Datos**
- **Riesgos de Control**

37

Riesgos Estructurales

- **Se presentan cuando dos instrucciones diferentes intentar usar el mismo recurso simultáneamente**
- **Ejemplo 1:**
 - Qué sucede si el procesador tiene una memoria de un puerto, compartida para datos e instrucciones?
 - Emplear memorias de datos e instrucciones independientes
 - En general, replicar los recursos necesarios para permitir todas las combinaciones posibles de instrucciones en el *pipeline*
- **Ejemplo 2:**
 - Qué sucede con los accesos al banco de registros?
 - Para resolver el riesgo estructural al momento de acceder al banco de registros, las escrituras se realizan en la primera mitad del ciclo, y las lecturas en la segunda

38

Riesgos

- Riesgos Estructurales
- Riesgos de Datos
- Riesgos de Control

39

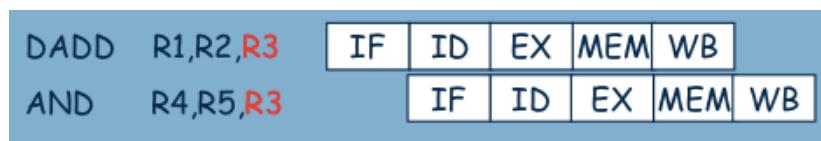
Riesgos de Datos

- Se producen entre dos instrucciones que utilizan el mismo operando
 - El operando puede ser registro o memoria
 - El problema es: ¿Cuándo los datos son modificados?
-
- RAR (Read After Read)
 1. RAW (Read After Write)
 2. WAR (Write After Read)
 3. WAW (Write After Write)

40

Tipos Riesgos Datos: (0) RAR

- **Read After Read**
- **Se producen en cualquier procesador**
- **NO es un problema**
- **NO se considera un riesgo**



Tipos Riesgos Datos: (1) RAW

- **Read After Write o “Dependencia Verdadera”**
- **Se producen en cualquier procesador**
- **SI es un problema**
- **Situación:**
 - Un operando es modificado para ser leído posteriormente.
 - Si la primera instrucción no ha terminado de escribir el operando, la segunda estará utilizando datos incorrectos.



Tipos Riesgos Datos: (2) WAR

- Write After Read o “Anti-Dependencia”
- Se producen en procesador fuera de orden
- Si es un problema
- Situación:
 - Leer un operando y escribir en él en poco tiempo.
 - Si la escritura finaliza antes que la lectura, la instrucción de lectura utilizará el nuevo valor y no el antiguo.

```
DADD R1,R2,R3
AND  R3,R4,R5
```

43

Tipos Riesgos Datos: (3) WAW

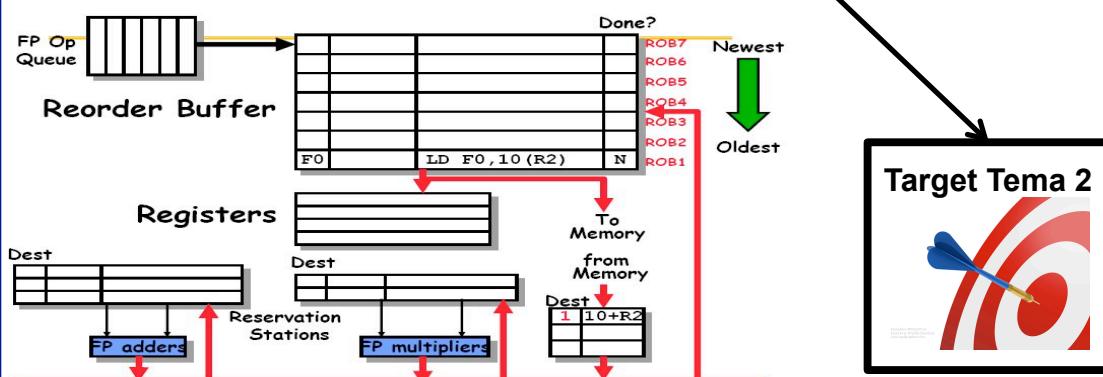
- Write After Write o “Dependencia de Salida”
- Se producen en procesadores:
 - Con ejecución fuera de orden ó
 - Con operaciones multiciclo
- Si es un problema
- Situación:
 - Dos instrucciones que escriben en un mismo operando.
 - La primera en ser emitida puede que finalice en segundo lugar, de modo que el operando final no tenga el valor adecuado.



44

Solución Riesgos WAW, WAR

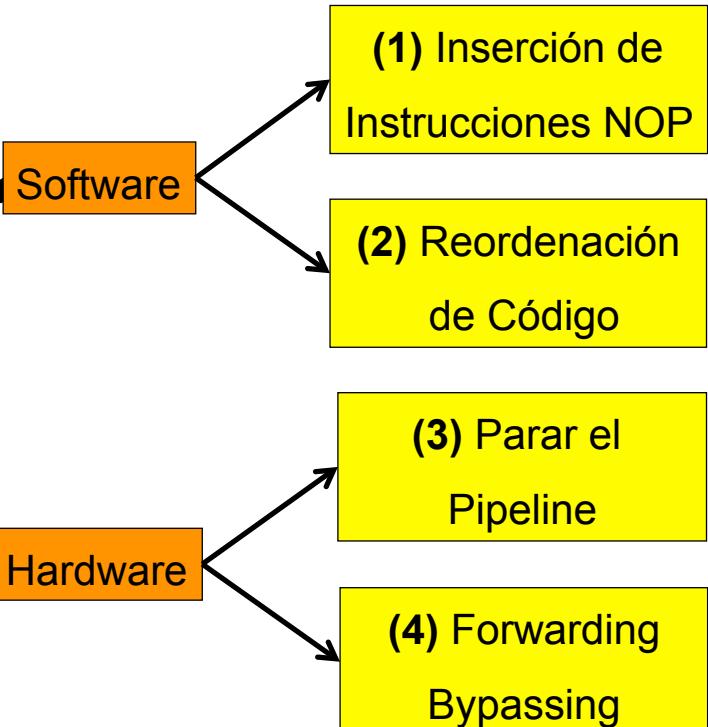
- Conocidos por DEPENDENCIAS DE NOMBRE
- Ocurren porque NÚMERO DE REGISTROS ES FINITO
- SOL 1: Reordenación de código → Target Tema 3
- SOL 2: Planificación dinámica insts.
 - Método del Marcador (Scoreboard)
 - Método de Tomasulo (Algoritmo de Tomasulo)



45

Solución Riesgos RAW

Solución
Riesgos RAW



46

Riesgos

- Riesgos Estructurales
- Riesgos de Datos
- **Riesgos de Control**

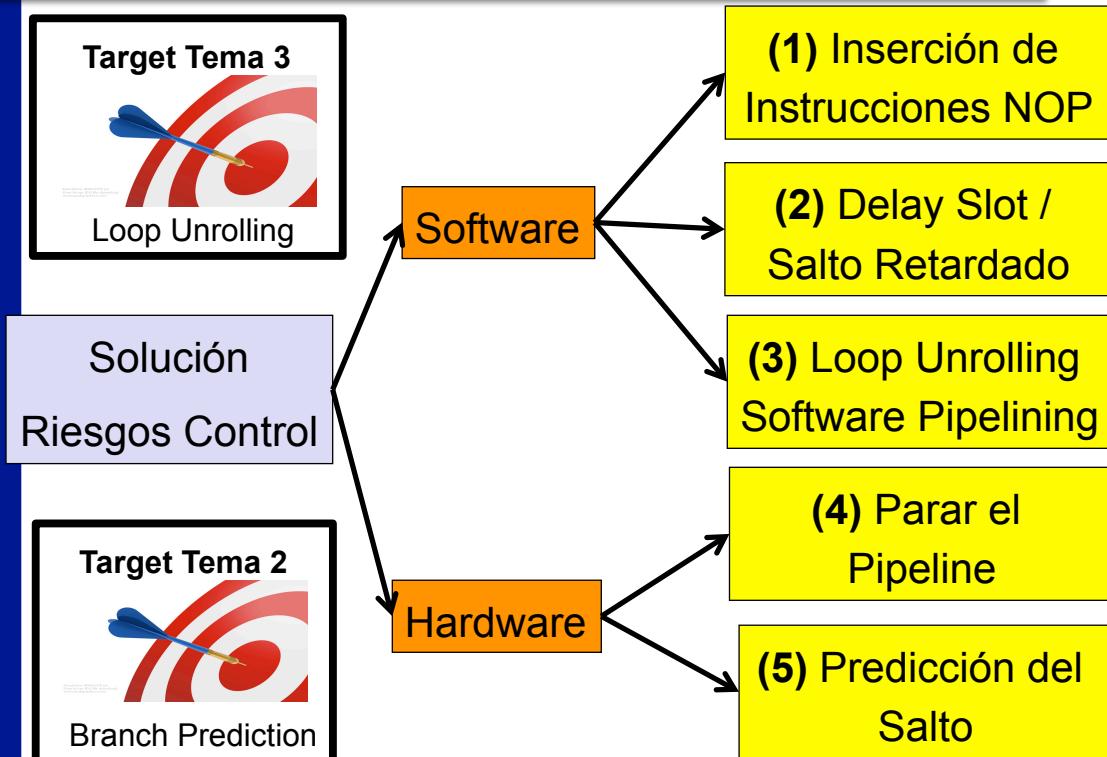
47

Riesgos de Control

- Surge de la necesidad de tomar una decisión basada en los resultados de una instrucción
- Salto determina el flujo de control
 - Siguiente instrucción depende del resultado del salto
 - Fetch de I tras salto debería hacerse en el siguiente ciclo de reloj
 - Pipeline no la sabe ya que está trabajando en la etapa ID del salto
- Necesario bloquear pipeline hasta conocer resultado
- **ATENCIÓN: 17% de las instrucciones MIPS ejecutadas son saltos**
 - Si no hacemos nada, el rendimiento puede verse muy limitado

48

Solución Riesgos Control



Índice Tema II

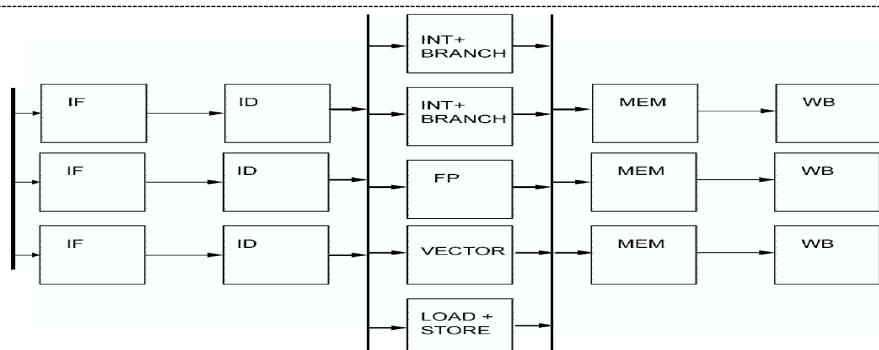
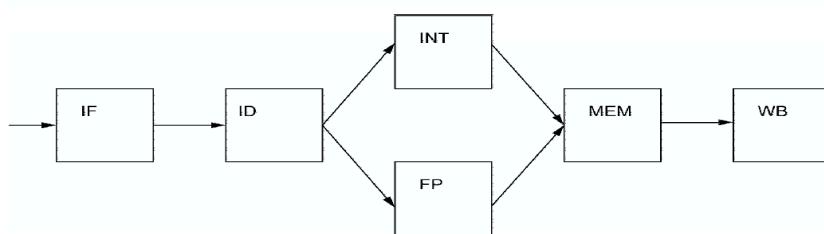
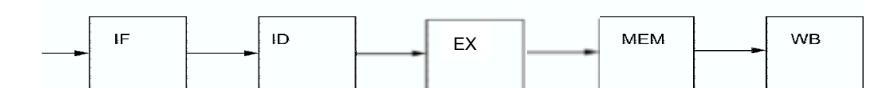
1. Introducción
2. Riesgos: Estructurales, Datos y Control
3. Camino de Datos del Procesador Superescalar
4. Técnicas Microarquitectónicas Avanzadas
5. Procesadores VLIW

Características Superescalar

- Término “superescalar” acuñado a finales de los 80s.
- Todas las CPUs modernas son superescalares.
- Es un desarrollo de la arquitectura con pipeline.
- Las etapas del pipeline se replican y en cada ciclo de reloj se emiten varias instrucciones.
- Se generan nuevos peligros por dependencias
- Hardware resuelve problemas en tiempo de ejecución
- Instrucciones entran/salen del procesador en orden
- Instrucciones pueden emitirse/ejecutarse en desorden

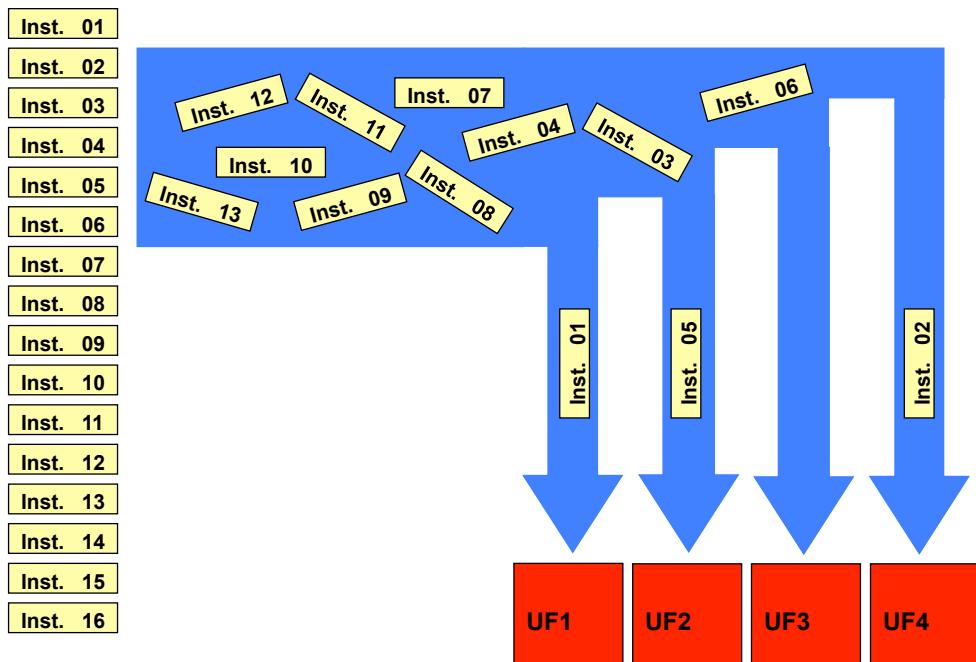
51

Evolución a Superescalar



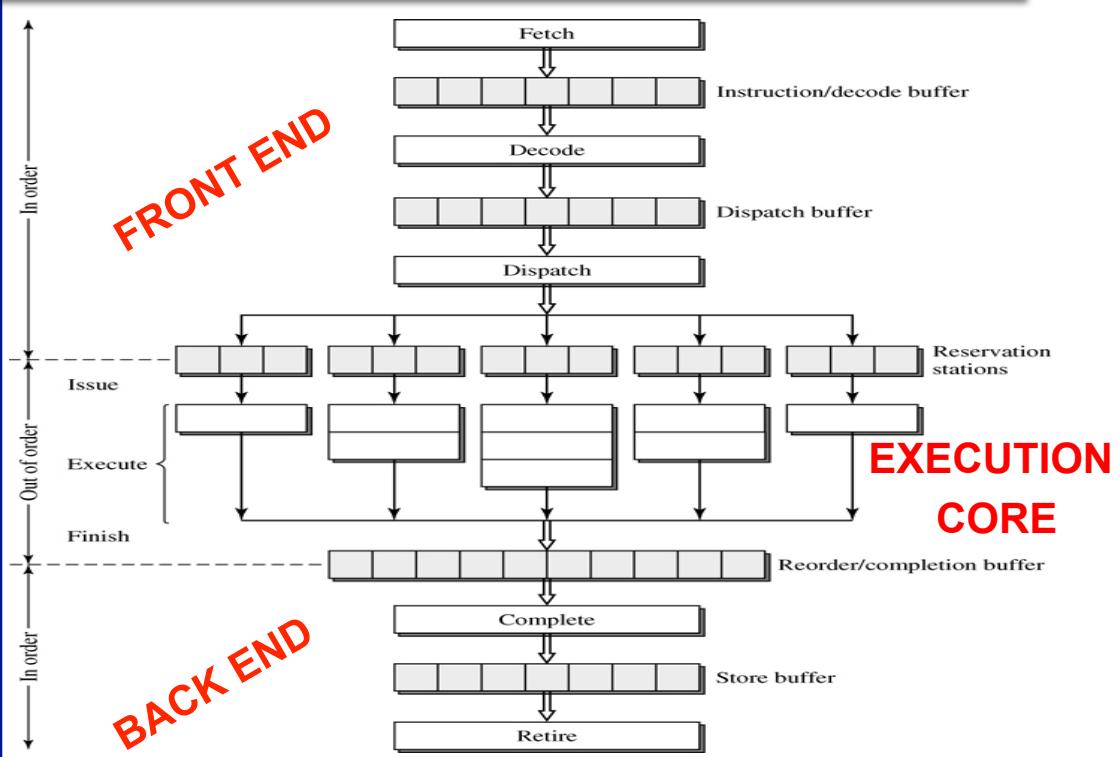
52

Visión General Simplificada



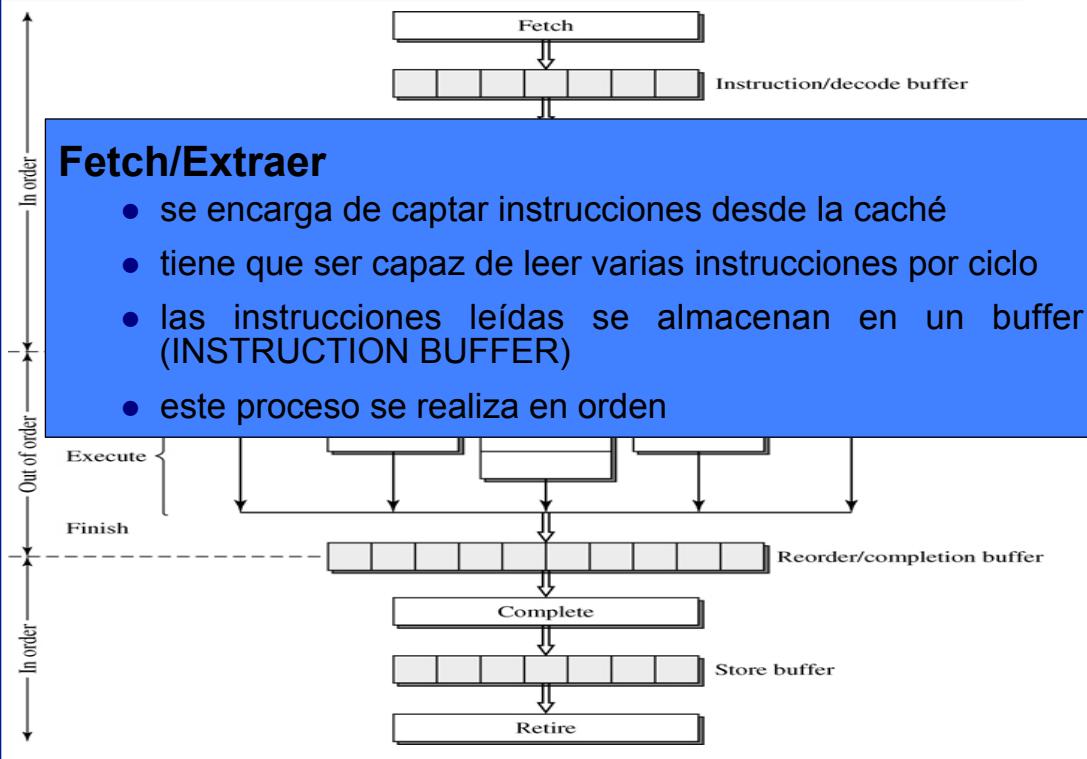
53

Visión General Extendida



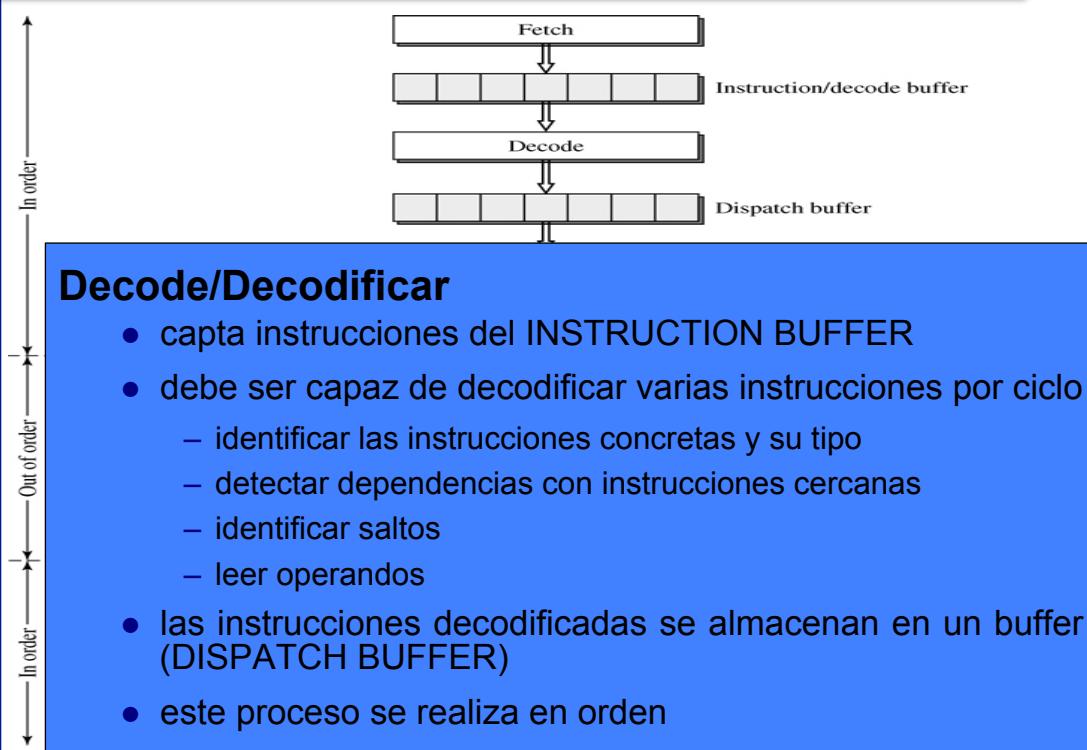
54

Etapa de Fetch



55

Etapa de Decode



56

Decode (Dependencias)

```
for (i=0; i<N; i++ {
    X[i] = Y[i] + Z[i];
    A[i] = B[i] * C[i];
}
```

Estático

r4	= M[Y]
r5	= M[Z]
r6	= r4+r5
M[X]	= r6
r7	= M[B]
r8	= M[C]
r9	= r7*r8
M[A]	= r9



r4	= M[Y]
r5	= M[Z]
r6	= r4+r5
M[X]	= r6
r4	= M[B]
r5	= M[C]
r6	= r4*r5
M[A]	= r6

Compilador no siempre puede evitar WAW y WAR

Decode (Dependencias)

- **Compilador no puede eliminar WAW y WAR porqué:**
 - número de registros es finito
 - no puede evaluar la dirección de los saltos (taken/ not taken)
- **En esta etapa (DINÁMICAMENTE) se deben detectar y eliminar todas las dependencias posibles**
 - RAW no se pueden evitar (se pueden suavizar)
 - WAR y WAR se pueden eliminar completamente
- **¿Cómo?**
 - Renombrar Registros Dinámicamente
- **Dos métodos de “Register Renaming”**
 - Banco de registros físicos
 - Reorder Buffer (ROB)

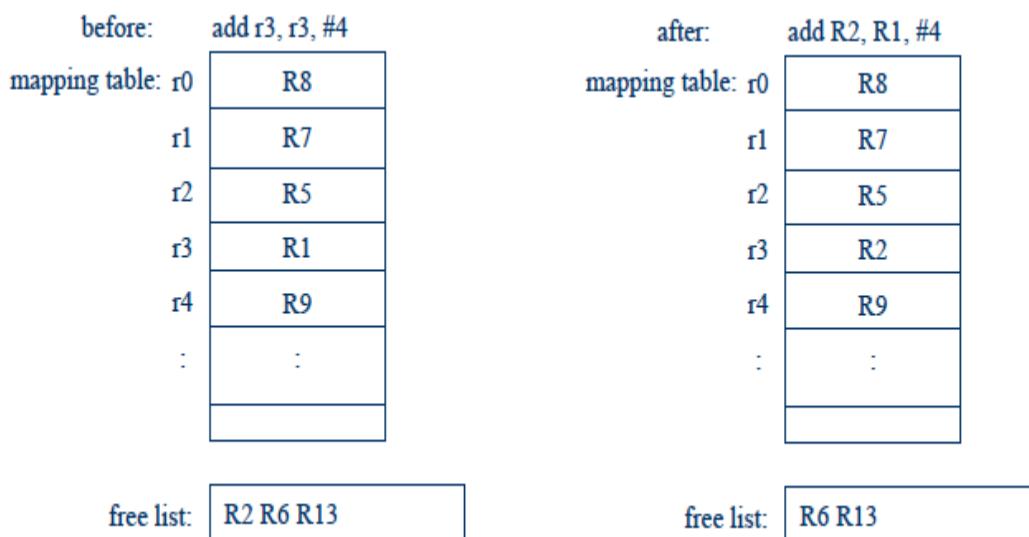
Renaming (Banco Registros Físicos)

- **Registros Lógicos**
 - los conocidos por programador y compilador
- **Registros Físicos**
 - conocidos sólo por el procesador
 - el número debe ser mayor al número de registros lógicos
- **Se utiliza:**
 - una tabla de mapeo para asociar registros lógicos con registros físicos
 - una lista de registros físicos libres
- **La decodificación y el renombramiento se realiza en orden por la etapa de decode & renaming**

59

Renaming (Banco Registros Físicos)

- **Ejemplo de renombramiento con registros físicos**

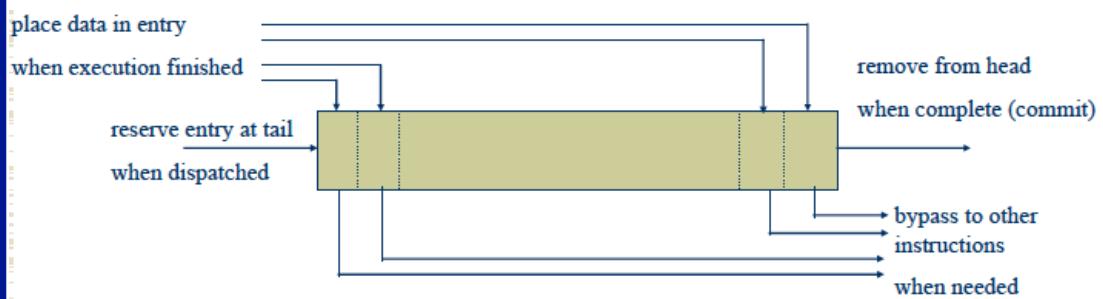


60

- Intel Pentium 4

Renaming (Reorder Buffer)

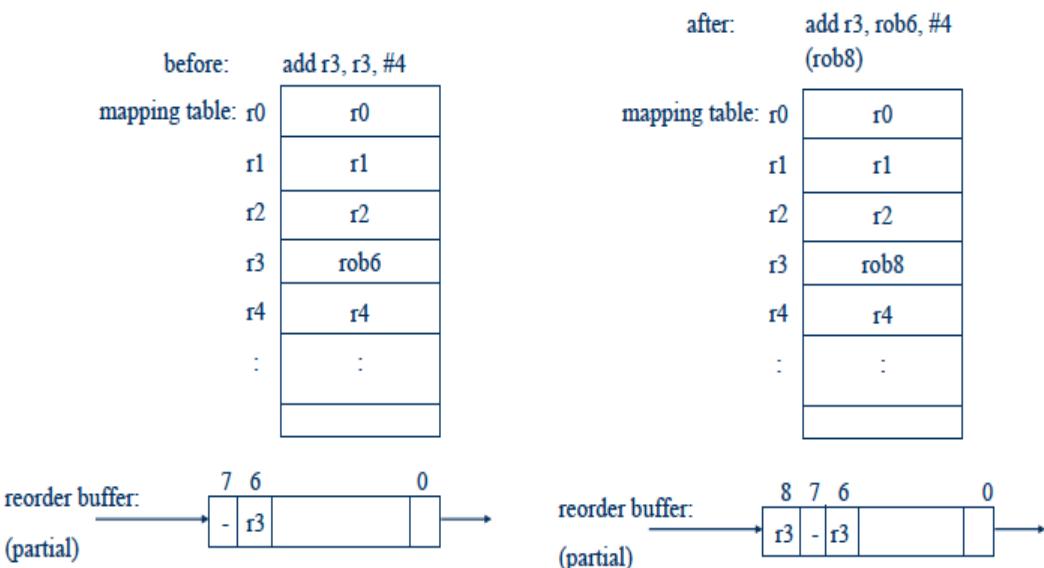
- La información de las instrucciones se introduce en ROB en orden (buffer es circular)
- Una instrucción se saca cuando ella y sus predecesoras han finalizado. En ese momento se actualizan los registros.
- Permite (1) renaming y (2) ejecución especulativa



61

Renaming (Reorder Buffer)

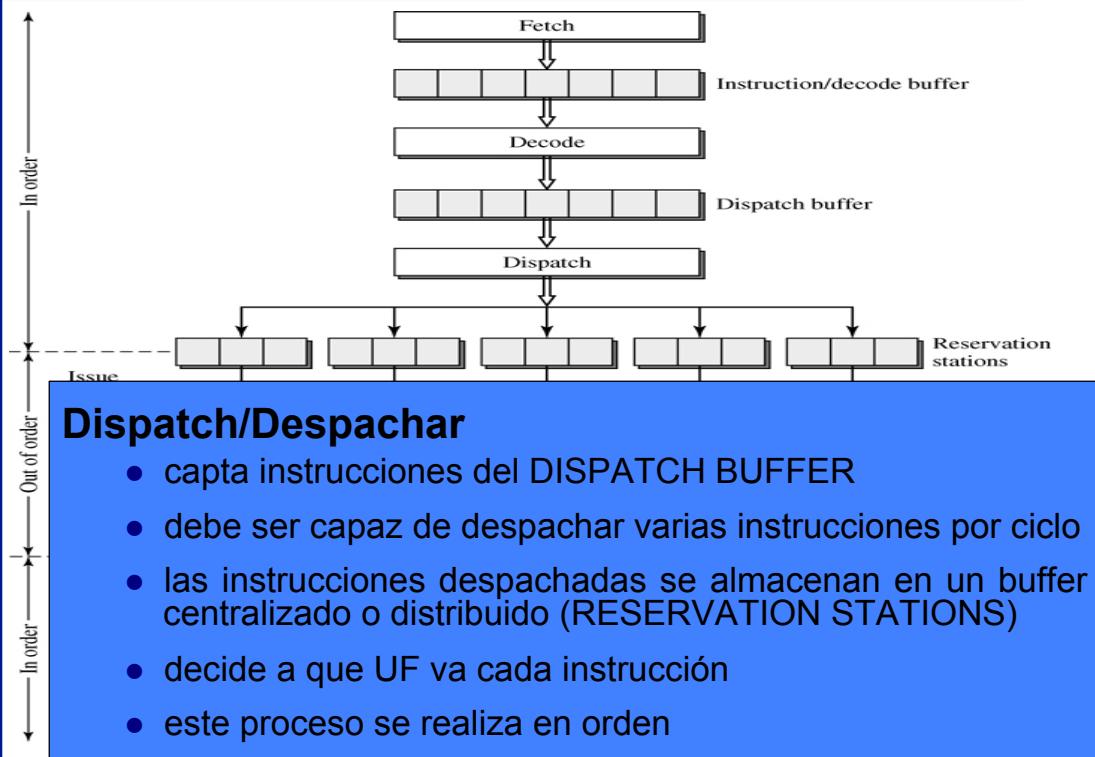
- Ejemplo de renombramiento con Reorder Buffer



62

■ Intel Core 2

Etapa de Dispatch

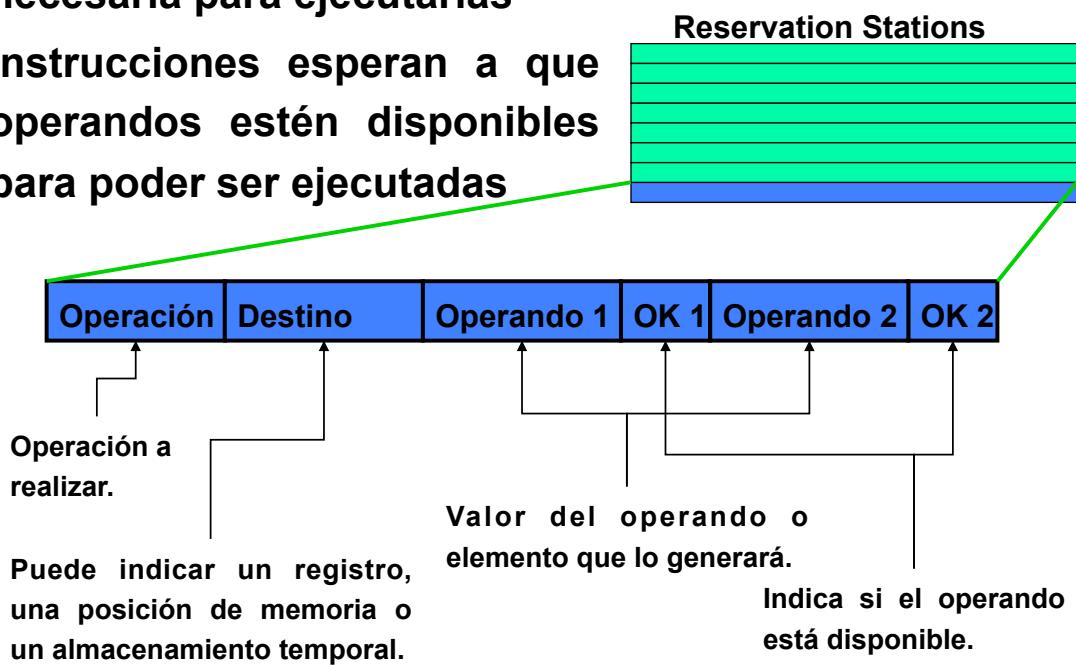


Dispatch (Reservation Stations)

- Nombre que toman los buffers tras el dispatch
- Hay dos enfoques
 - Centralizado: 1 buffer para todas las UFs
 - Distribuido: 1 buffer para cada UF
- 1 estación de reserva tiene espacio para 1 instrucción
- Instrucciones que entran en las RS están libres de:
 - dependencias de control: branch prediction
 - dependencias de nombre (WAR,WAW): register renaming
- Reservation Station puede unificarse con ROB

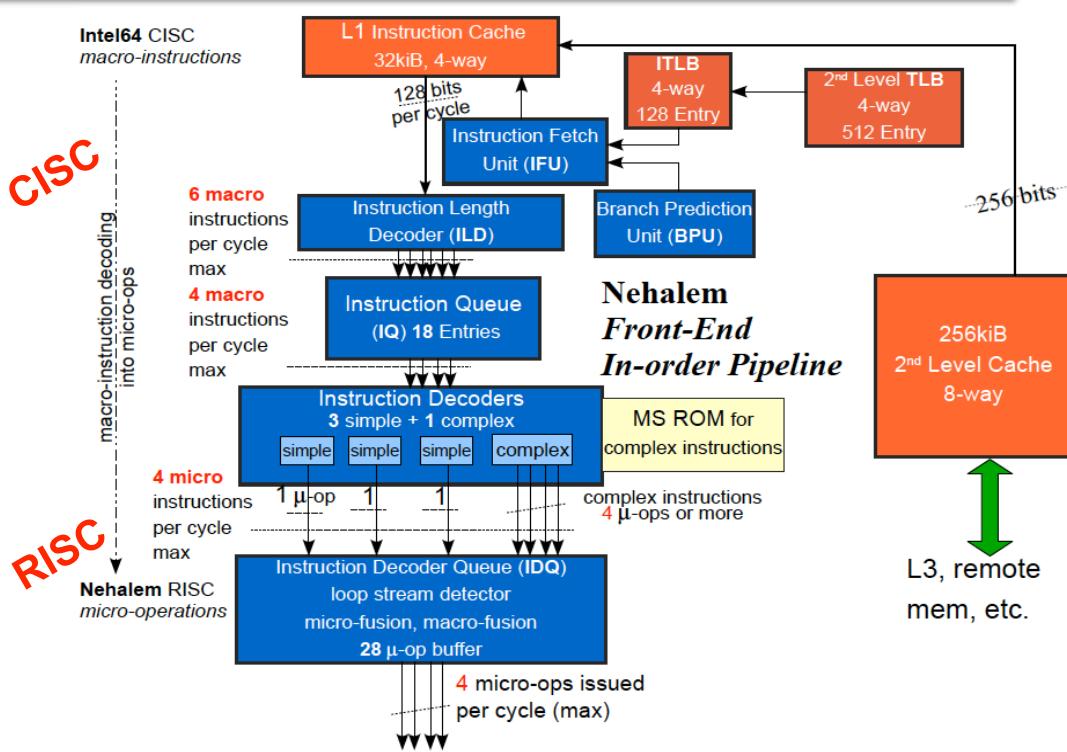
Dispatch (Reservation Stations)

- No se guardan instrucciones sino la información necesaria para ejecutarlas
- Instrucciones esperan a que operandos estén disponibles para poder ser ejecutadas



65

Intel Nehalem (Front End)

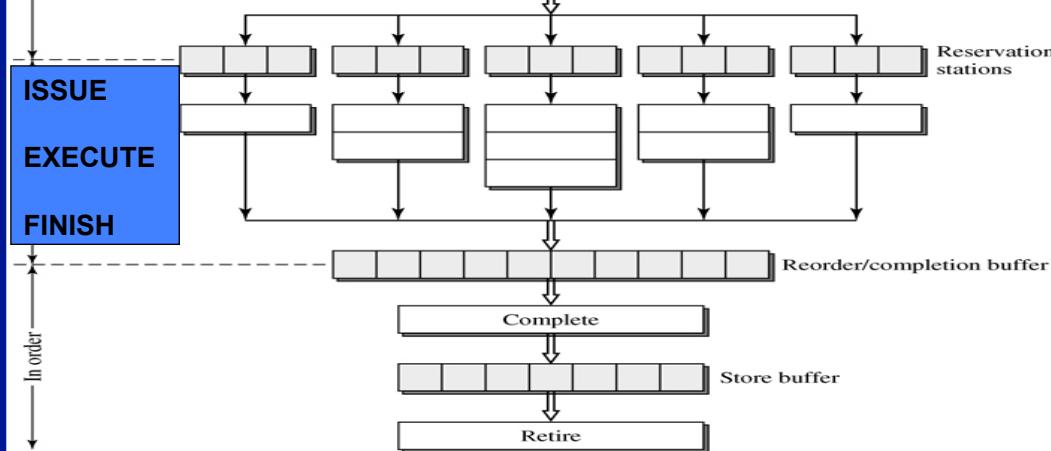


66

Etapa de Execute

Execute/Ejecutar

- capta instrucciones de las RESERVATION STATIONS
- las instrucciones ejecutadas se almacenan en el REORDER BUFFER
- este proceso se puede realizar en orden o fuera de orden



67

Execute (Issue, Exec, Finish)

■ Issue/Emitir

- una instrucción se **emite** cuando sus operandos están disponibles
- una lógica de wake-up se encarga de “despertar” la instrucción y de insertarla en la unidad funcional correspondiente

■ Execute/Ejecutar

- una instrucción se **ejecuta** dentro de una unidad funcional
- cada unidad funcional tiene una determinada latencia

■ Finish/Finalizar

- tras la ejecución se almacena la instrucción en el reorder buffer
- el reorder buffer tiene las instrucciones ordenadas

■ Scheduling.

- En que orden se ejecutan las instrucciones. Hay dos opciones:
 - Estático: cuando el hardware no tiene control (in order execution)
 - Dinámico: cuando el hardware puede reordenar (O-O-O execution)

68

Clasificación Superescalar

Common name	Issue structure	Hazard detection	Scheduling	Distinguishing characteristic	Examples
Superscalar (static)	dynamic	hardware	static	in-order execution	mostly in the embedded space: MIPS and ARM
Superscalar (dynamic)	dynamic	hardware	dynamic	some out-of-order execution, but no speculation	none at the present
Superscalar (speculative)	dynamic	hardware	dynamic with speculation	out-of-order execution with speculation	Pentium 4, MIPS R12K, IBM Power5
VLIW/LIW	static	primarily software	static	all hazards determined and indicated by compiler (often implicitly)	most examples are in the embedded space, such as the TI C6x
EPIC	primarily static	primarily software	mostly static	all hazards determined and indicated explicitly by the compiler	Itanium

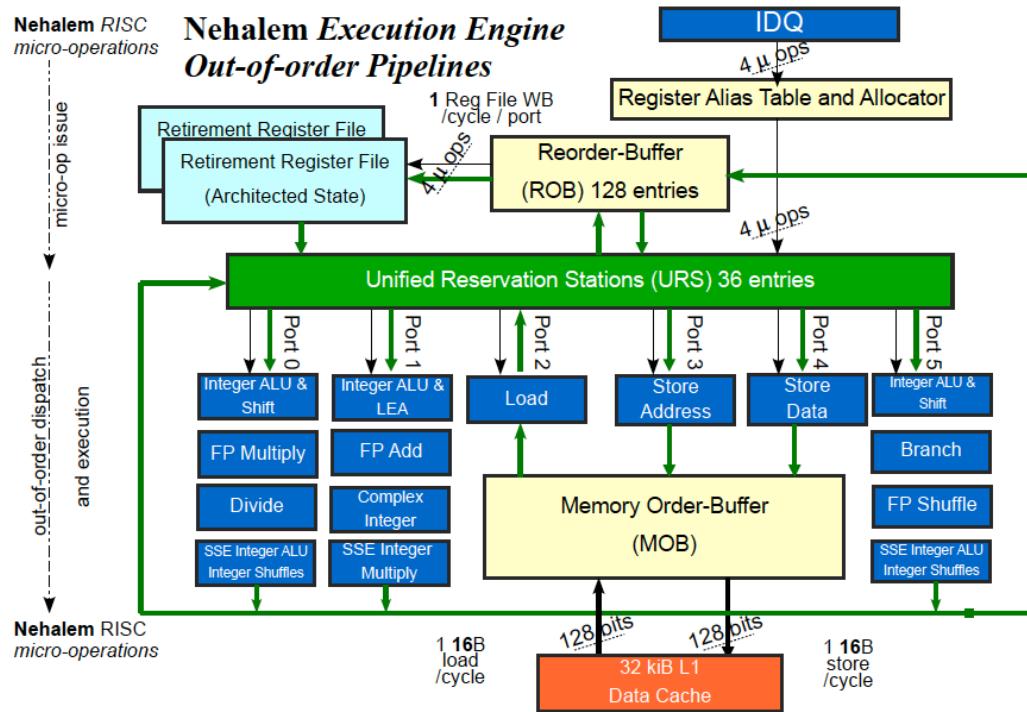
69

Ventana de Instrucciones

- **Instruction Window (IW)**
- **Concepto que indica el número máximo de instrucciones que puede haber en el pipeline**
- **Tam Reorder Buffer (ROB) = Instruction Window (IW)**
- **Reservation Stations (RS)**
 - Centralizadas o distribuidas
 - Tamaño RS <= Tamaño ROB
 - RS centralizada puede estar unificada con ROB
- **Se habla de IW como estructura cuando ROB y RS están unificados**

70

Intel Nehalem (Execution Core)



Etapa de Complete y Commit

Complete/Completar y Commit/Cometer

- captar instrucciones desde el REORDER BUFFER
- debe ser capaz de completar varias instrucciones por ciclo
- las instrucciones completadas (sólo stores) se almacenan en un buffer (STORE BUFFER)
- este proceso se realiza en orden
- necesario banco de registros con varios puertos de escritura

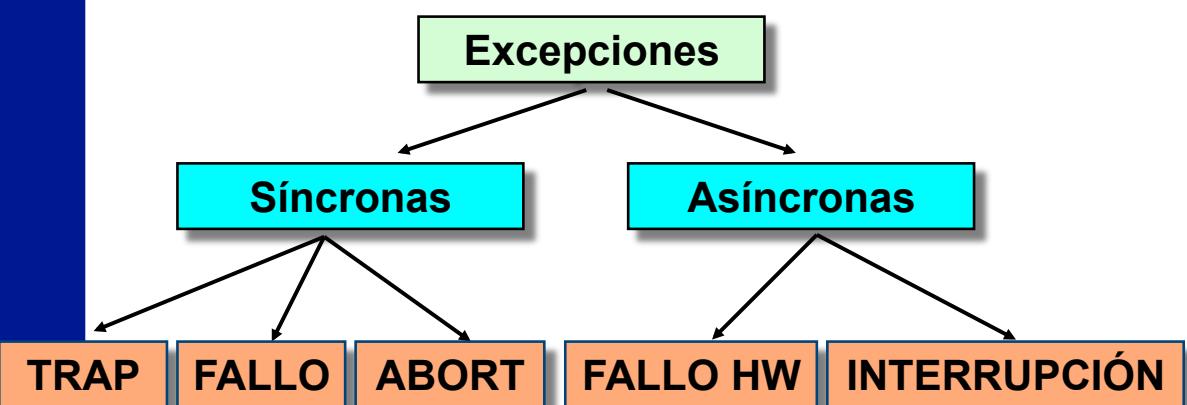
Etapa de Complete y Commit

- **COMPLETE o WRITE BACK**
 - resultados de la instrucción se escriben en banco de registros
- **COMMIT**
 - Una a una y EN ORDEN se sacan las instrucciones del reorder buffer
 - Si instrucción del ROB no ha finalizado se para el COMMIT
- **En esta etapa se validan las especulaciones**
 - predicción de saltos
 - predicción de memoria (memory disambiguation)
- **Si se detecta error, se para pipeline y se vacía todo**
- **Las instrucciones de store se almacenan en un buffer FIFO (STORE BUFFER)**

73

Excepciones

- Sucesos extraordinarios que requieren atención especial por parte del procesador



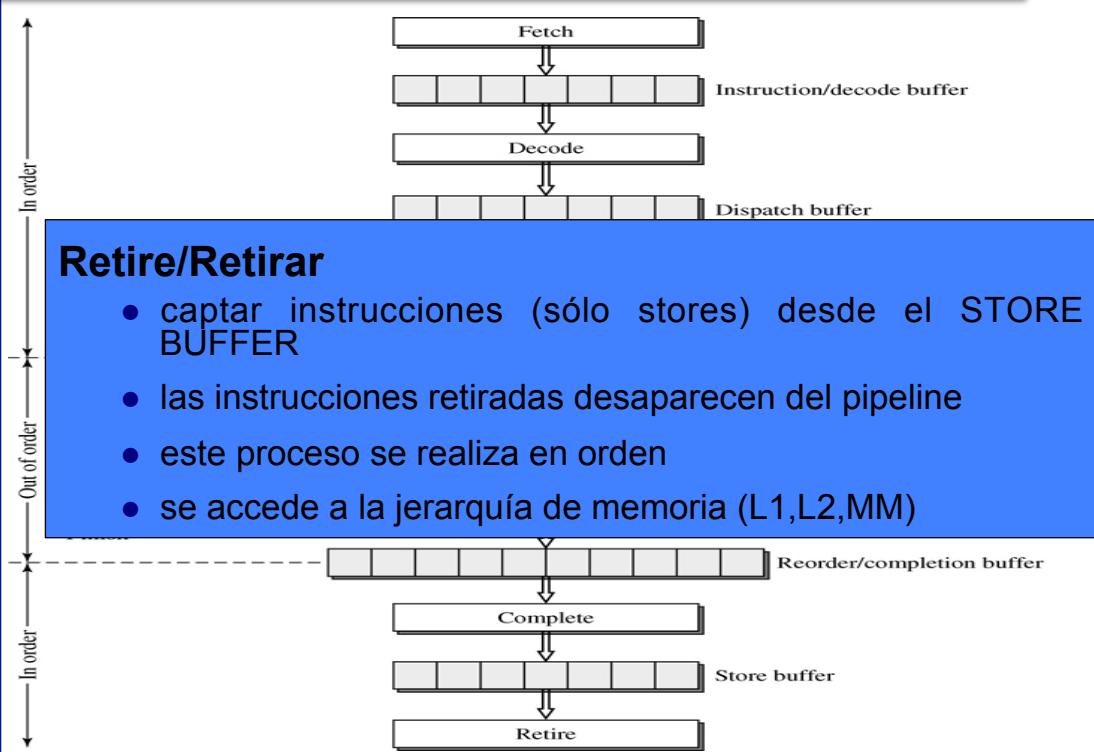
dispositivo E/S preparado para E/S

Excepciones Síncronas

- También conocidas como excepciones PRECISAS
- Producidas por una instrucción determinada, es necesario ejecutar un código asociado
- Procesador “in order” no son un problema
- Procesador O-O-O debe mantener consistencia:
 - ROB es fundamental
 - se detecta la instrucción que produce la excepción
 - instrucciones anteriores en ROB se acabarán ejecutando y se les realizará el commit
 - instrucciones posteriores del ROB se descartarán
 - posteriormente el procesador ejecutará el código asociado a esa excepción

75

Etapa de Retire



76

Índice Tema I

1. Introducción
2. Riesgos: Estructurales, Datos y Control
3. Camino de Datos del Procesador Superescalar
- 4. Técnicas Microarquitectónicas Avanzadas**

5. Procesadores VLIW

77

Técnicas Microarquitectónicas

1. Prefetching (FETCH)
2. Branch Prediction(FETCH)
3. Register Renaming (DECODE)
4. Multiple Issue Estático/Dinámico (ISSUE, EXECUTE)
5. Memory Disambiguation (ISSUE, EXECUTE)
6. Multithreading (FETCH, DECODE, ISSUE, EXEC, etc)

78

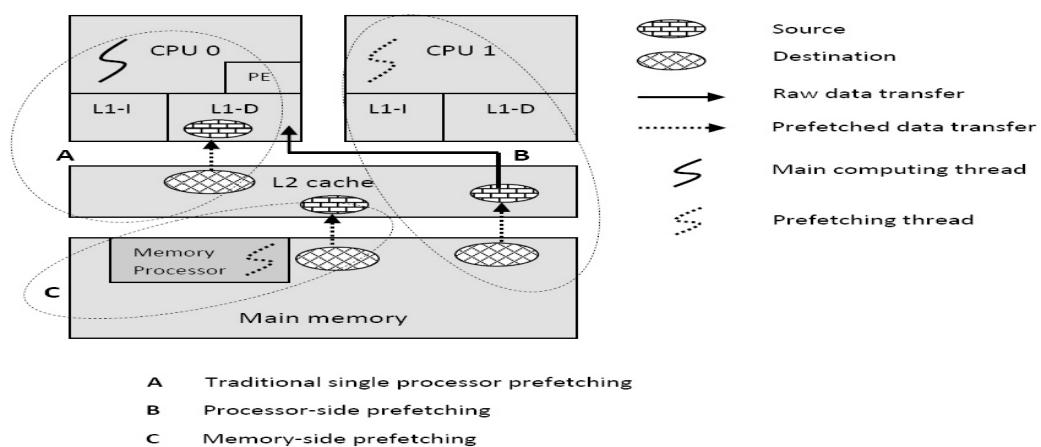
Speculative Execution

- Adelantar trabajo que puede ser innecesario o erróneo
- Branch Prediction 
- Prefetching 
- Speculative Memory Disambiguation 
- Value Prediction 
- Thread Speculation 
- Slipstream Processors 
- Necesarios mecanismos de recuperación
- La penalización en caso de fallo es muy costosa

79

(1) Prefetching

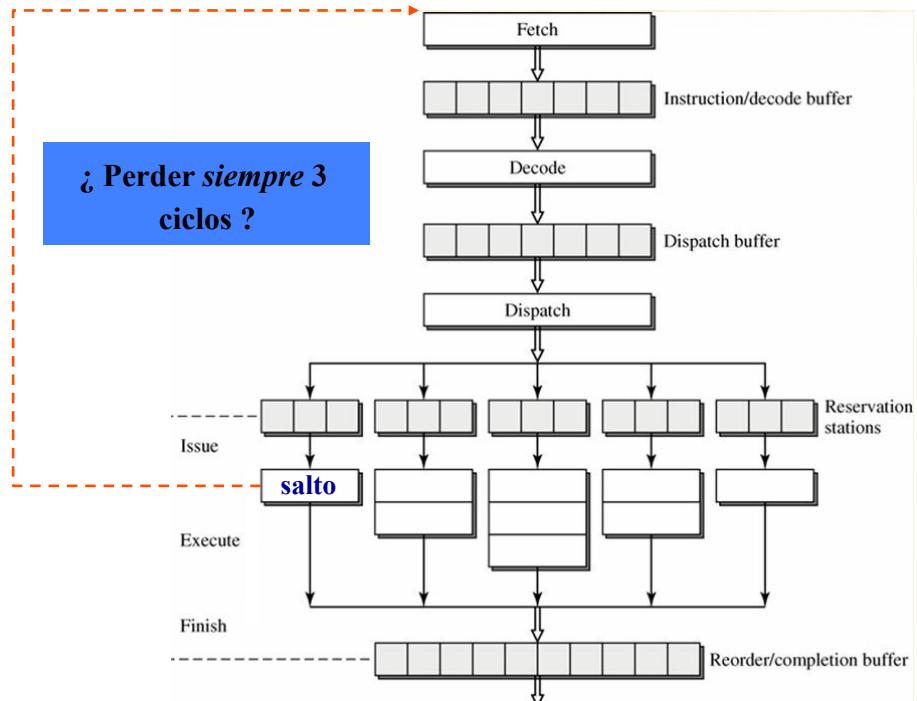
- Técnica ESPECULATIVA
- No es necesario mecanismo de recuperación
- Se intenta traer a memoria caché datos en previsión de que puedan ser usados en un futuro cercano



80

(2) Predicción de Saltos

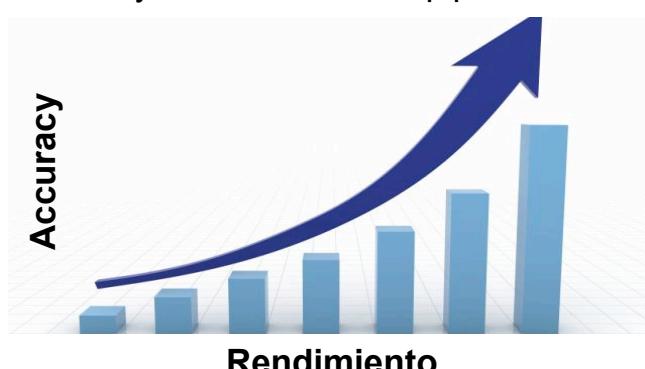
¿ Perder siempre 3 ciclos ?



81

(2) Predicción de Saltos

- Parar el pipeline limita excesivamente el rendimiento
- Se predice camino que tomará salto y se ejecuta
- Técnica ESPECULATIVA
- Si hay error en la predicción:
 - necesario volver al estado de antes del salto
 - vaciar y volver a llenar el pipeline es costoso



Muy importante
para el
rendimiento fallar
poco

82

(2) Predicción de Saltos

- Se realiza en la etapa de FETCH
- Estructuras especiales
 - BHT: Branch History Table
 - almacena información del comportamiento pasado
 - BTB: Branch Target Buffer
 - almacena la dirección de destino del salto tomado
- Tipos de Predictores
 - Estáticos
 - se determina “outcome” en tiempo de compilación
 - Dinámicos
 - Last Value
 - Stride
 - Correlation

83

(3) Register Renaming

- Elimina serialización causada por reuso de registros
- Estaticamente
 - se encarga el compilador (no siempre los puede resolver)
- Dinámicamente
 - se encarga procesador en tiempo de ejecución
 - existen dos métodos fundamentalmente
 - BANCO DE REGISTROS FÍSICOS
 - REORDER BUFFER

84

```

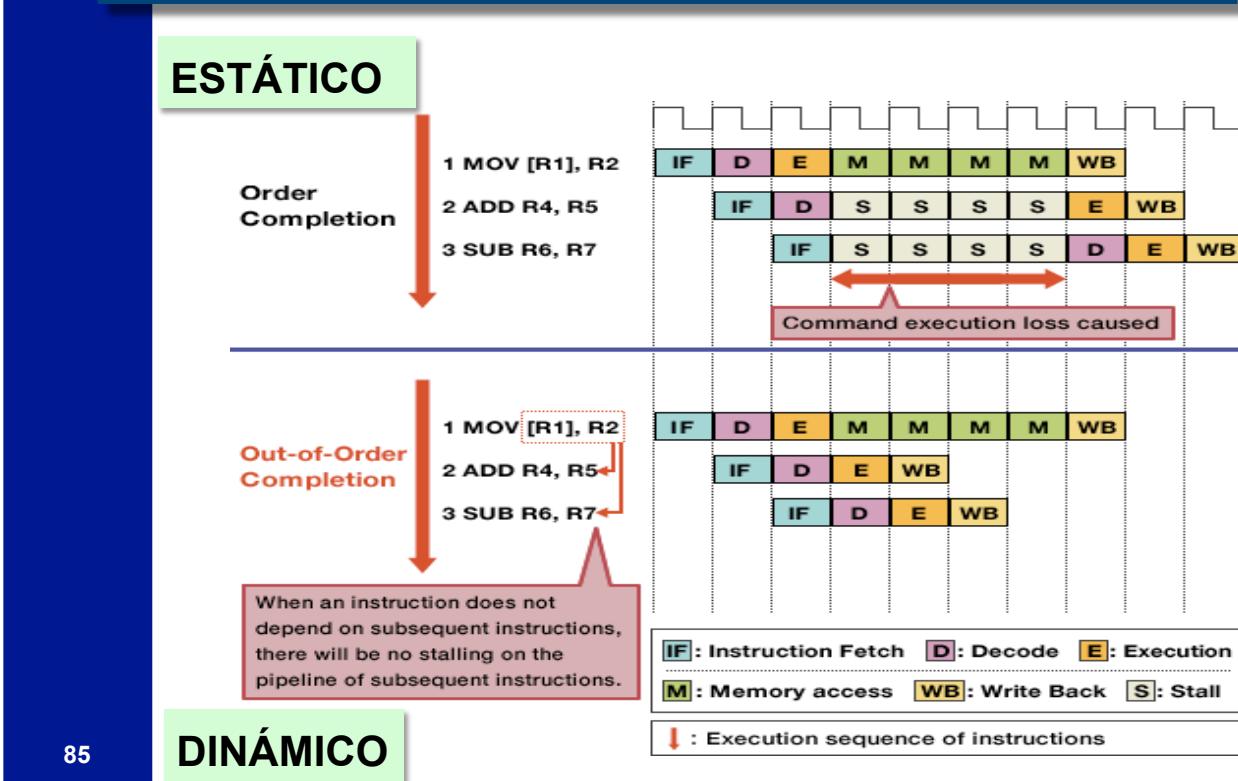
1. R1=M[1024]
2. R1=R1+2
3. M[1032]=R1
4. R1=M[2048]
5. R1=R1+4
6. M[2056]=R1
  
```



```

1. R1=M[1024]
2. R1=R1+2
3. M[1032]=R1
4. R2=M[2048]
5. R2=R2+4
6. M[2056]=R2
  
```

(4) Multiple Issue Estático/Dinámico



85

(4) Multiple Issue Dinámico

- Permite la ejecución fuera de orden (OOO)
- Existen dos métodos fundamentales
 - Método del Marcador (Scoreboard)
 - Método de Tomasulo (Algoritmo de Tomasulo)

Scoreboard
CDC 6600 (1964)



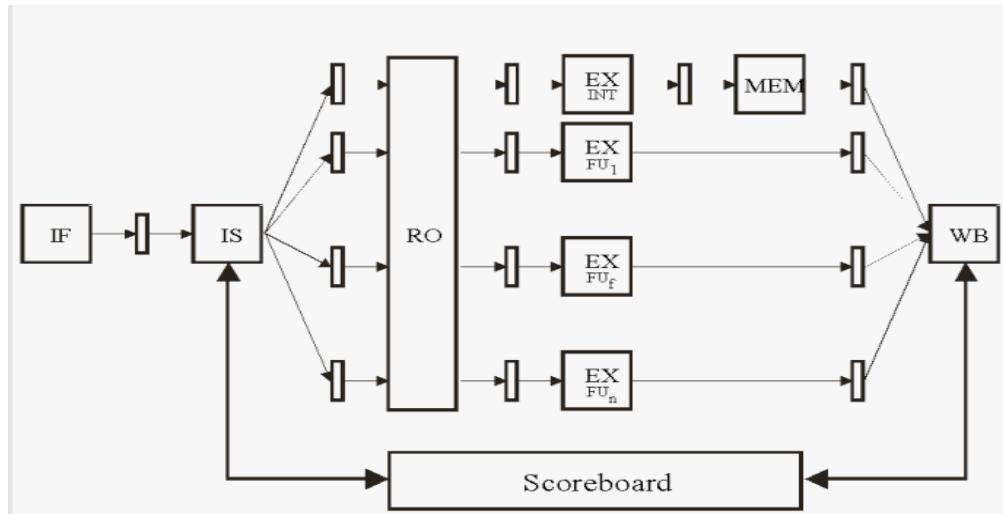
Tomasulo
IBM 360/91 (1964)



86

(4) Mult.Issue.Din: Scoreboarding

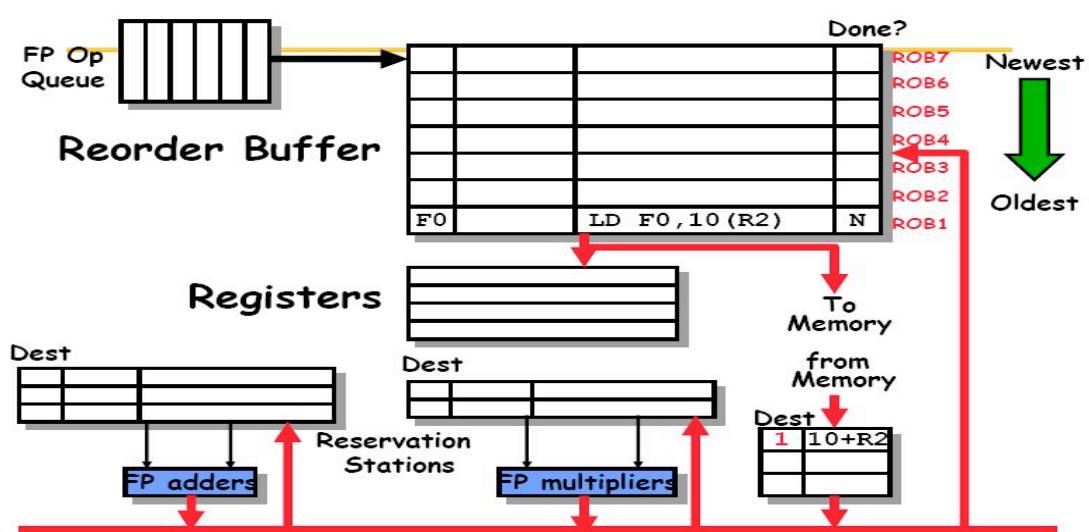
- NO resuelve en tiempo de ejecución las dependencias WAW y WAR
- En caso de WAR y WAW: stall (se para el procesador)



87

(4) Mult.Issue.Din: Tomasulo

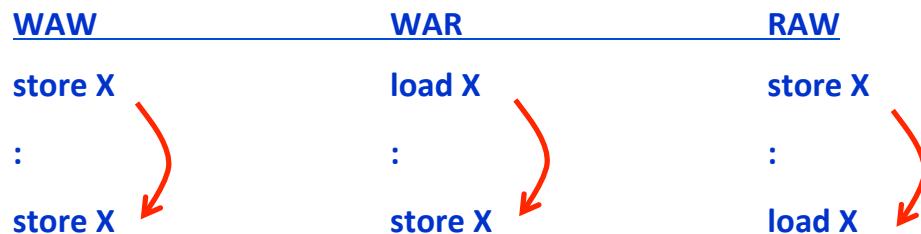
- SI Resuelve en tiempo de ejecución las dependencias WAW y WAR mediante el renombrado de registros
- Mayoría procs actuales implementan alguna variante



88

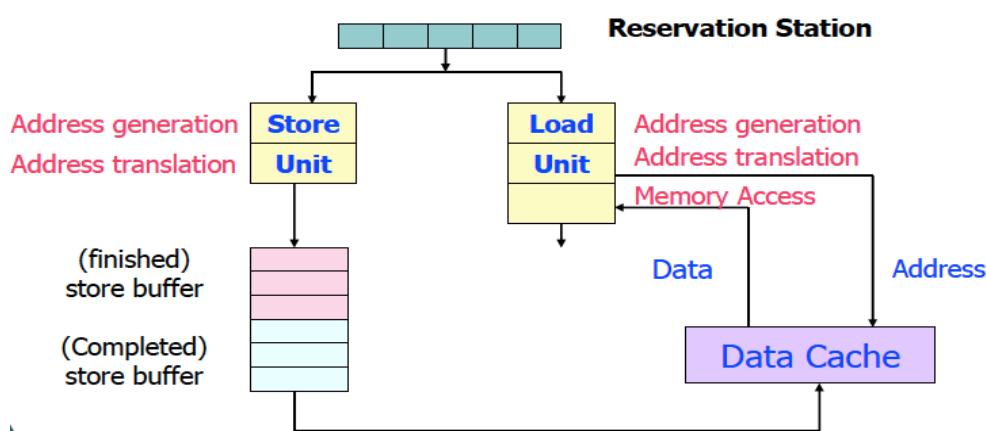
(5) Memory Disambiguation

- Latencias de memoria son un gran problema
- MEJORA: ejecutar operaciones de memoria O-O-O



- A. Total Order of Loads/Stores
- B. Load Bypassing (not speculative)
- C. Load Bypassing (speculative)
- ⁸⁹ D. Load Forwarding

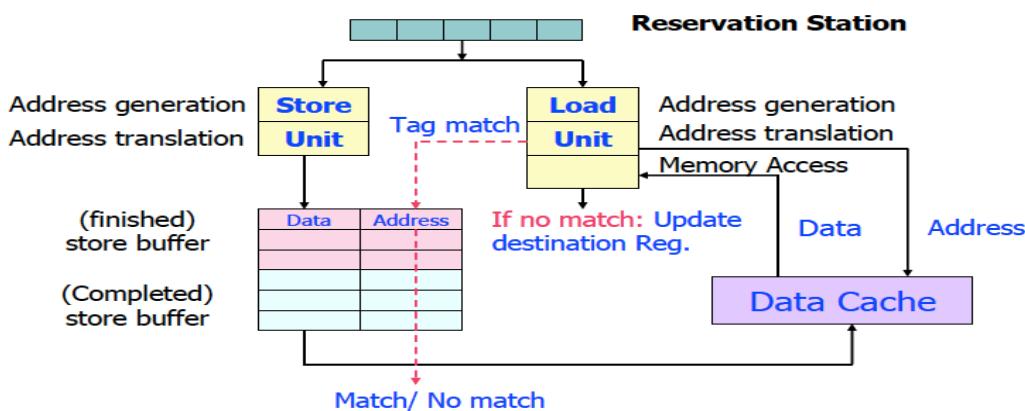
(5.a) Total Order



- Se mantienen todos los Loads y Stores EN ORDEN
- Loads y Stores pueden ejecutarse fuera de orden con respecto al resto de instrucciones
- Rendimiento es menor

(5.b) Load Bypassing

- Stores se mantienen en orden
- Loads pueden adelantar a los Stores si:
 - direcciones de los stores previos son conocidas
 - load que adelanta no coincide su @ con la de un store previo
- Hardware adicional pero mayor rendimiento



91

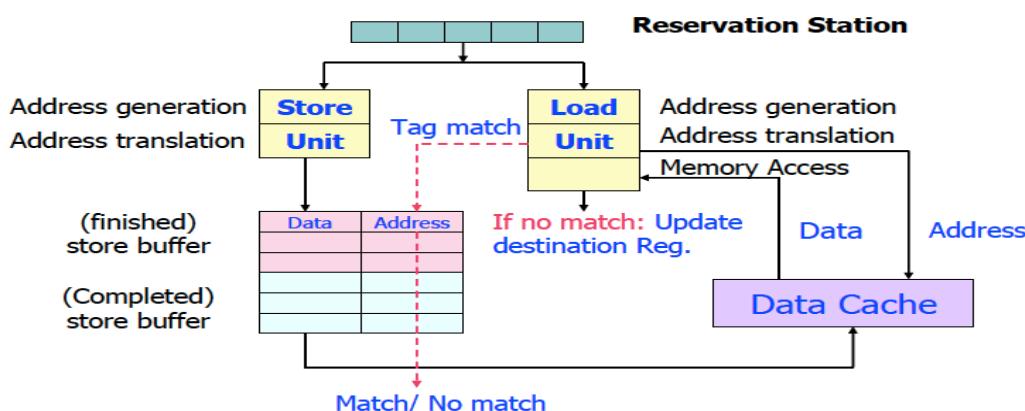
(5.c) Speculative Load Bypassing

- Stores se mantienen en orden
- Loads pueden adelantar a los Stores sin necesidad de conocer las @ de los stores previos
- Obviamente si hay coincidencia, load no adelanta
- Técnica ESPECULATIVA
- Necesarios mecanismos de recuperación para fallos
- La clave está en acertar a ejecutar los Loads que no dependen de Stores anteriores
- Se utilizan distintos predictores para ello

92

(5.d) Load Forwarding

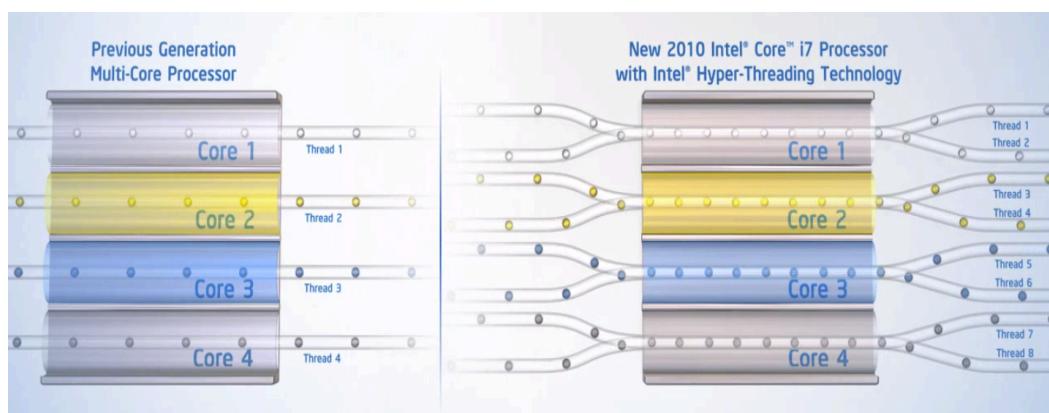
- Ortogonal con Load Bypassing
- IDEA: si Load tiene dependencia con Store aún presente en el Store Buffer, se toma el dato de allí (en lugar de ir a memoria caché)



93

(6) Multithreading

- Varios hilos de ejecución simultáneos en misma CPU



- Intel Core I7: 2 threads x core (2tx6c)
- IBM Power 3: 4 threads x core (4tx8c)
- SUN UltraSparct T3: 8 threads x core (8tx16c)



94

Índice Tema I

1. Introducción
2. Riesgos: Estructurales, Datos y Control
3. Camino de Datos del Procesador Superescalar
4. Técnicas Microarquitectónicas Avanzadas
5. Procesadores VLIW

95

VLIW

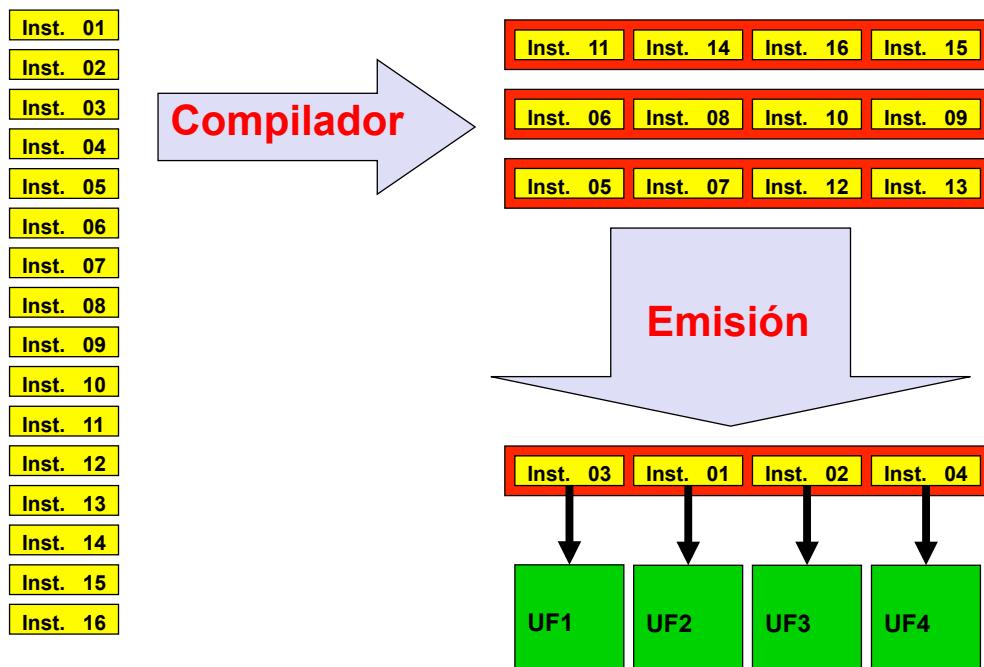
- **Very Long Instruction Word**
- **Implementa paralelismo a nivel de instrucción**
- **Compilador fundamental para determinar paralelismo**
 - empaqueta varias operaciones en una nueva instrucción
 - cada campo de nueva instrucc. especializado en UF determinada
 - se encarga de empaquetar correctamente instrucciones evitando los problemas (dependencias, disponibilidad del hardware, etc...)
 - En cada ciclo se emite sólo una de las nuevas instrucciones.
- **Juego de instrucciones muy simples en cuanto a número pero grandes en cuanto al tamaño**

Proc. Superescalar:
Múltiple Issue Dinámico

Procesador VLIW:
Múltiple Issue Estático

96

VLIW



97

VLIW

■ Ventajas

- Hardware más simple ya que el compilador se encarga de un gran número de tareas.
- Mayor número de unidades funcionales ya que queda libre mucho espacio en el chip.
- Menor consumo de potencia

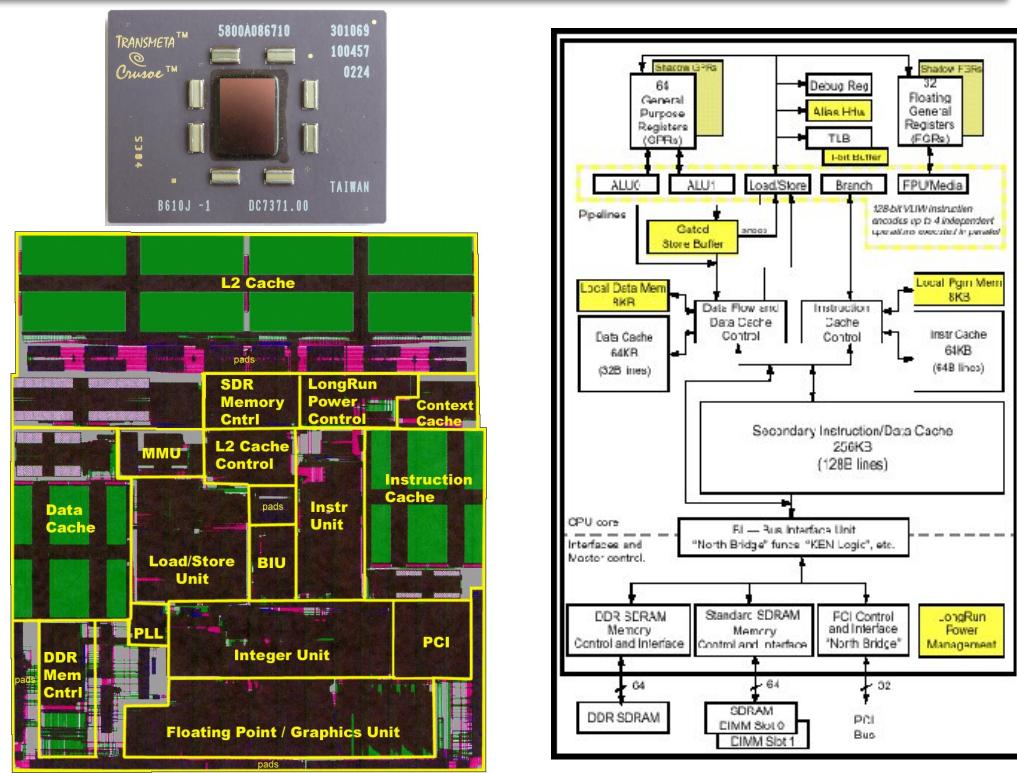
■ Inconvenientes

- Complejidad del compilador al tenerse que ocupar de numerosos asuntos.
- Dificultad para llenar todos los campos de las instrucciones.
- Mayor tamaño del código debido al problema anterior.
- Incompatibilidad del código de cara a versiones con nuevas distribuciones de las unidades funcionales.

98

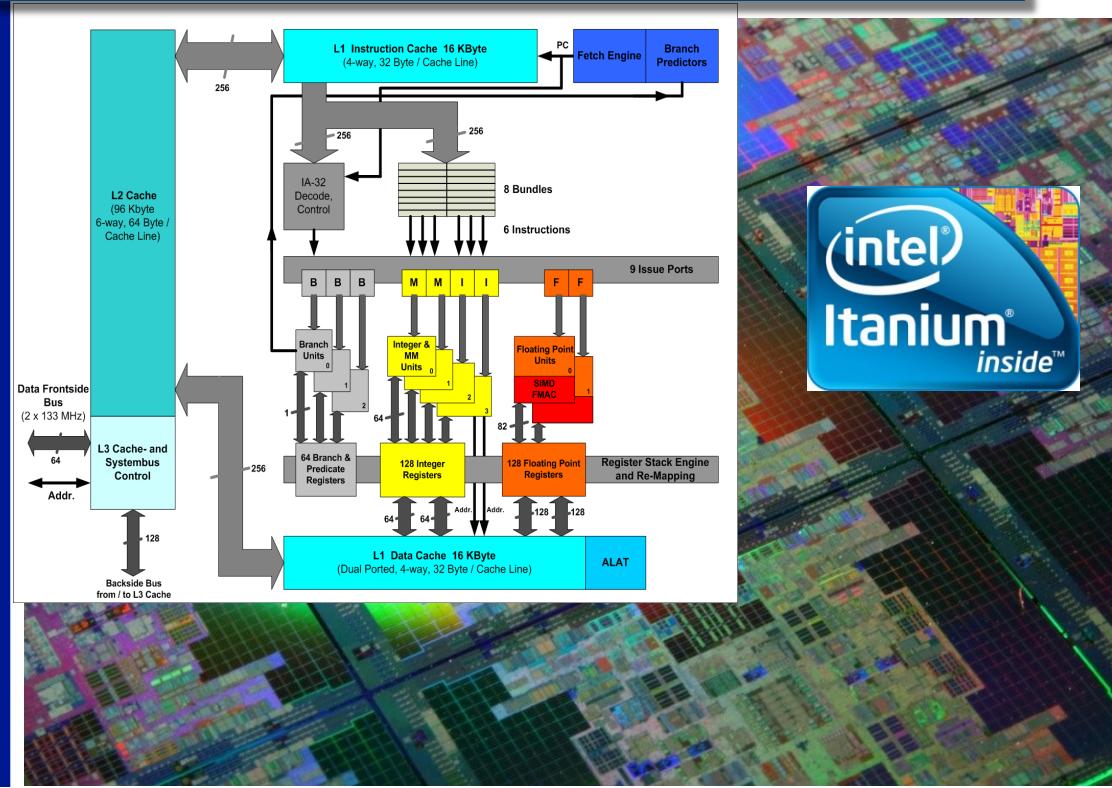
Transmeta Crusoe

99



99

Intel Itanium IA-64 EPIC



100

Fin Tema I

Te creo que tu CPU necesita refrigeración extra pero, ¿podrías hacer espacio para la comida?

