

PROCESADORES DE EMISIÓN MÚLTIPLE

- ❖ ¿Cómo conseguir un $CPI < 1$ ($IPC > 1$)?
 - Emitiendo varias instrucciones en un mismo ciclo de reloj
- ❖ Dos clases de procesadores de emisión múltiple
 - Superescalares y VLIW (*Very Long Instruction Word*)
 - Característica común: poseen **varias unidades de ejecución** capaces de funcionar en paralelo

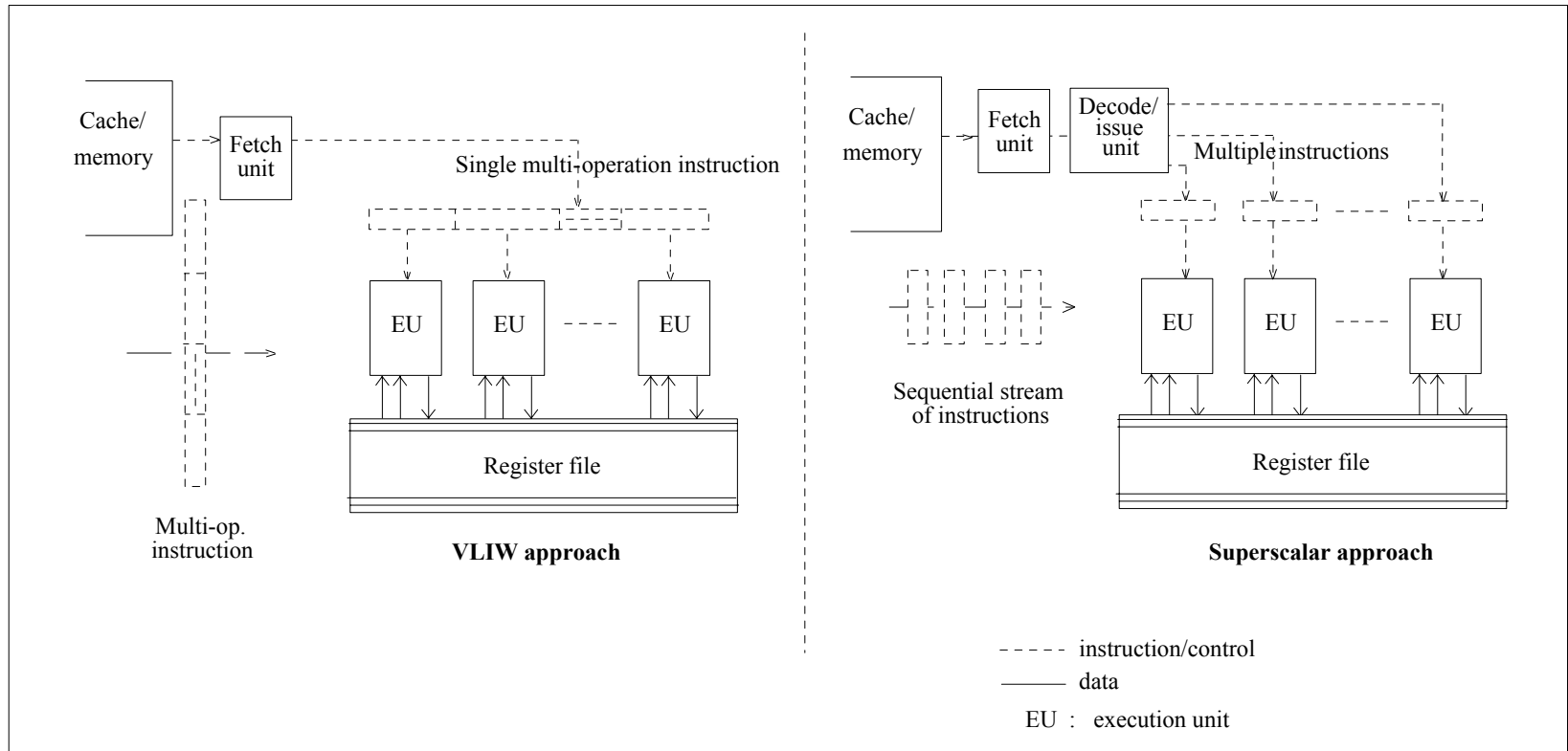
Procesadores superescalares

- ❖ Descodifican un número pequeño de instrucciones (entre 2 y 6) en cada ciclo de reloj
 - Si es posible, se emitirán en un mismo ciclo
- ❖ Planificación instrucciones:
 - Típicamente hardware (mediante técnicas basadas en el *scoreboard* o el algoritmo de Tomasulo)
 - Parte de la planificación suele hacerla el compilador
- ❖ Gran implantación en los procesadores comerciales

Procesadores VLIW

- ❖ Cada instrucción codifica varias operaciones (varias instrucciones en un procesador convencional)
 - Las instrucciones tienen entre 100 bits y 1Kbit
- ❖ Las operaciones de una instrucción se ejecutarán en paralelo en las distintas unidades de ejecución
- ❖ Precisan compiladores específicos
 - Planificación estática
- ❖ Entre 5 y 30 unidades de ejecución

Organización superescalares/VLIW..



DLX superescalar..

- ❖ Emisión de dos instrucciones por ciclo
 - Una de operación de punto flotante, la otra entera (*load, store, branch*, operación entera)
 - Más sencillo que permitir la emisión de dos instrucciones de cualquier tipo
- ❖ Fetch: 64 bits en cada ciclo de reloj
- ❖ Descodificación simple aconseja:
 - Par de instrucciones: <entera, punto flotante>

DLX superescalar..

❖ Ejecución ideal en DLX superescalar

Tipo instrucción

Etapas

Entera

F D X M W

Punto flotante

F D X M W

Entera

F D X M W

Punto flotante

F D X M W

Entera

F D X M W

Punto flotante

F D X M W

DLX superescalar..

- ❖ DLX superescalar requiere una considerable cantidad de operaciones FP en los programas para que la mejora de rendimiento sobre el DLX escalar sea sustancial
- ❖ En la tabla anterior no se muestra la mayor duración de las operaciones FP en EXE
- ❖ Para mantener la frecuencia de emisión de instrucciones FP es preciso:
 - Tener suficientes unidades de ejecución de FP y/o
 - Encauzar las unidades de ejecución de FP
 - ✓ En la práctica se hacen ambas cosas

DLX superescalar..

- ❖ La instrucción entera usa diferentes registros y diferentes unidades funcionales que la FP =>
 - Sólo hay **colisión** cuando la instrucción *entera* es *load*, *store* o *move* sobre registros FP
 - ✓ ¿Cómo resolver la colisión?
 - Emitir solas las instrucciones enteras con operandos de punto flotante (sencillo, pero bajo rendimiento)
 - Añadir un puerto adicional de lectura y otro de escritura al banco de registros FP

DLX superescalar..

- ❖ Puede aparecer una dependencia RAW entre la instrucción entera y la FP
 - En ese caso sólo se emite la entera
 - El resto de dependencias son las mismas que en el procesador escalar
- ❖ Retardos de carga y de salto:
 - un ciclo que ahora supone **las tres siguientes instrucciones**

DLX superescalar..

❖ Planificación estática del bucle

```
Loop:    LD      F0, 0 (R1)
          ADDDD  F4, F0, F2
          SD      0 (R1), F4
          SUBI   R1, R1, 8
          BNEZ   R1, Loop
```

suponiendo las mismas latencias del DLX escalar:

(FP ALU Op, FPU ALU Op)	→ 3
(FP ALU Op, Store Double)	→ 2
(Load Double, FP ALU Op)	→ 1

DLX superescalar..

	<i>Instrucción entera</i>	<i>Instrucción FP</i>	<i>Ciclo reloj</i>
Loop:	LD F0, 0 (R1)		1
	LD F6, -8 (R1)		2
	LD F10, -16 (R1)	ADDD F4, F0, F2	3
	LD F14, -24 (R1)	ADDD F8, F6, F2	4
	LD F18, -32 (R1)	ADDD F12, F10, F2	5
	SD 0 (R1), F4	ADDD F16, F14, F2	6
	SD -8 (R1), F8	ADDD F20, F18, F2	7
	SD -16 (R1), F12		8
	SUBI R1, R1, 40		9
	SD 16 (R1), F16		10
	BNEZ R1, Loop		11
	SD 8 (R1), F20		12

DLX superescalar..

- ❖ Obsérvense las dependencias entre el primer LD y el primer ADDD y entre éste y el primer SD
 - Ello **exige** planificar el bucle y **desenrollarlo cuatro veces** para eliminar paradas (ciclos en los que no se emite ninguna instrucción)
 - $12\text{ciclos}/5 = \mathbf{2,4 \text{ ciclos}}$ por elemento del vector
 - ✓ Recordemos: $14\text{ciclos}/4 = 3,5$ ciclos por elemento, en DLX escalar (planificando y desenrollando el bucle)
 - ✓ La **mejora** ha sido de **1,5 veces**

Planificación en procesadores VLIW

- ❖ El compilador utiliza diversas técnicas
 - Desenrollamiento de bucles, planificación de trazas...
- ❖ Supongamos
 - DLX VLIW con capacidad de emitir:
 - ✓ 2 referencias a memoria, 2 operaciones FP y una operación entera o bifurcación
 - Ignoramos el retardo de salto
- ❖ ¿Cómo queda nuestro bucle una vez planificado y desenrollado?

Desenrollamiento de bucles en DLX VLIW

<i>Referencia a memoria 1</i>	<i>Referencia a memoria 2</i>	<i>Operación FP 1</i>	<i>Operación FP 2</i>	<i>Op. entera/ branch</i>
<i>ld f0,0(r1)</i>	<i>ld f6,-8(r1)</i>			
<i>ld f10,-16(r1)</i>	<i>ld f14,-24(r1)</i>			
<i>ld f18,-32(r1)</i>	<i>ld f22,-40(r1)</i>	<i>add f4,f0,f2</i>	<i>add f8,f6,f2</i>	
<i>ld f26,-48(r1)</i>		<i>add f12,f10,f2</i>	<i>add f16,f14,f2</i>	
		<i>add f20,f18,f2</i>	<i>add f24,f22,f2</i>	
<i>sd 0(r1),f4</i>	<i>sd -8(r1),f8</i>	<i>add f28,f26,f2</i>		
<i>sd -16(r1),f12</i>	<i>sd -24(r1),f16</i>			<i>subi r1,r1,56</i>
<i>sd 24(r1),f20</i>	<i>sd 16(r1),f24</i>			
<i>sd 8(r1),f28</i>				<i>bnez r1,loop</i>

- 9 ciclos/7 (copias del bucle) = **1.29** ciclos por elemento
- Se necesitan más registros FP

Comparación superescalares/VLIW..

- ❖ Planificación estática => hardware más sencillo.
Por tanto los VLIW admiten
 - Un **mayor número de unidades de ejecución**
 - Una **frecuencia de reloj mayor** (principal ventaja)
- ❖ Compatibilidad con procesadores secuenciales
 - Superescalares: sí. VLIW: no
- ❖ Posibilidad de programar en ensamblador
 - Superescalares: sí. VLIW: no (hay que considerar muchos aspectos para formar una instrucción)

Comparación superescalares/VLIW..

- ❖ VLIW: dependencia del compilador de las características tecnológicas del procesador
 - Para una planificación eficiente, el compilador debe tener en cuenta detalles tecnológicos (por ejemplo, latencias de la unidades de ejecución)
 - ✓ Un compilador no puede usarse en procesadores nuevos (con nuevas características tecnológicas) aunque sean compatibles en el sentido convencional
- ❖ En los VLIW los fallos de caché provocan la parada completa del procesador (no así en los superescalares)

Comparación superescalares/VLIW

- ❖ Debido a las dependencias, a menudo, no es posible empaquetar en una instrucción VLIW todas las operaciones que es capaz de codificar =>
 - Consumo adicional de memoria
 - ✓ Se puede comprimir la instrucción en memoria y expandirla al llevarla a caché o al decodificarla
 - Pérdida de rendimiento (en el tiempo de lectura y ejecución de una instrucción VLIW se leen y ejecutan menos operaciones de las teóricamente posibles)

Limitación del paralelismo en procesadores superescalares y VLIW..

- ❖ (Estas limitaciones afectan antes a los VLIW porque tienen más unidades de ejecución)
- ❖ Cuanto mayor sea el número N de unidades de ejecución
 - Mayor dificultad para encontrar un conjunto de N operaciones entre las que no haya dependencias de datos
 - (..)

Limitación del paralelismo en procesadores superescalares y VLIW

- La frecuencia de saltos y bifurcaciones por cada emisión de instrucción/instrucciones es mayor
 - ✓ Los saltos y bifurcaciones perjudican el paralelismo
- En definitiva, mayor dificultad para tener ocupadas todas las unidades de ejecución
- Se necesitan bancos de registros multipuerto y memorias multipuerto o entrelazadas para que varias operaciones puedan acceder a ellos a la vez