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

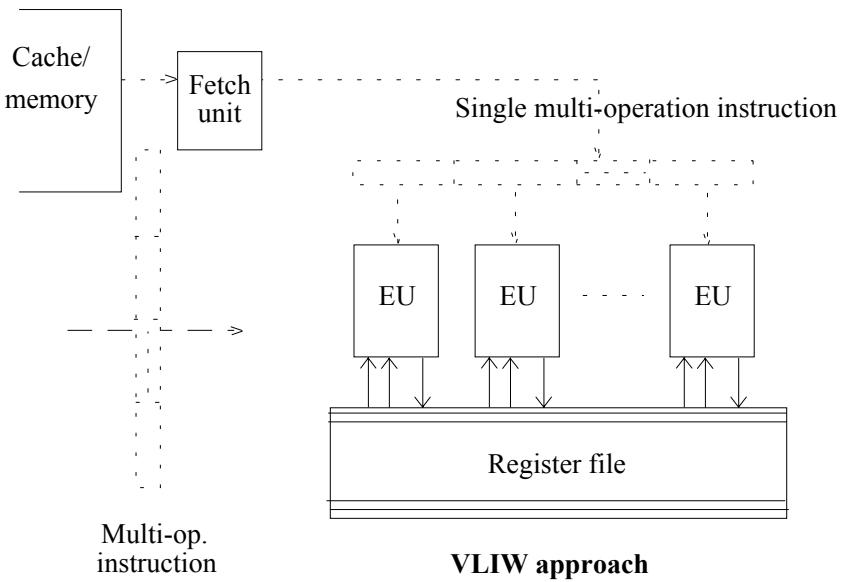
Procesadores VLIW

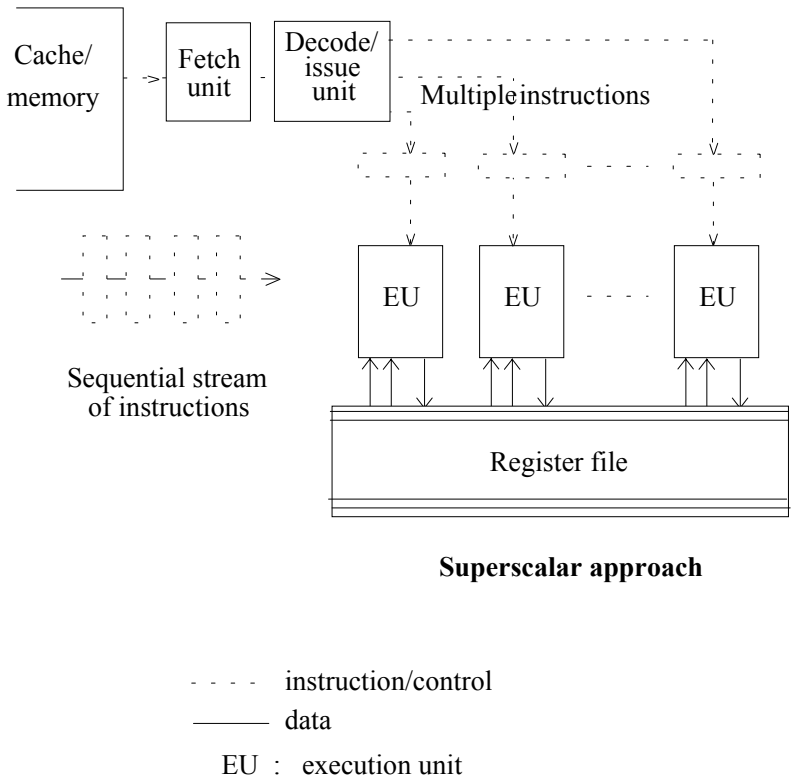
- Cada instrucción codifica varias operaciones (varias instrucciones en un procesador convencional)
  - Las instrucciones tienen entre 100 bits y 1Kbit
- Las **operaciones de una instrucción** se ejecutarán **en paralelo** en las distintas unidades de ejecución
  - Por eso también se denominan **EPIC** (*Explicitly Parallel Instruction Computers*)
- Precisan compiladores específicos
  - Planificación estática
- Entre 5 y 30 unidades de ejecución

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

Organización VLIW/superescalares..





Planificación en procesadores VLIW..

- El compilador utiliza diversas técnicas
  - Desenrollamiento de bucles, planificación de trazas...
- Supongamos
  - MIPS VLIW con capacidad de emitir:
    - ✓2 referencias a memoria, 2 operaciones FP y una operación entera o bifurcación
  - Ignoramos el retardo de salto
- ...

Planificación en procesadores VLIW

- ¿Cómo queda el siguiente bucle una vez planificado y desenrollado?

```
Loop:   L.D      F0, 0 (R1)
        ADD.D    F4, F0, F2
        S.D      F4, 0 (R1)
        DADDUI   R1, R1, #-8
        BNE      R1, R2, Loop
```

Se suponen las mismas latencias del MIPS escalar:

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

Desenrollamiento del bucle en MIPS VLIW

Referencia a memoria 1	*Referencia a memoria 2*	*Operación FP 1*	*Operación FP 2*	*Op. entera/ branch*
l.d f0,0(r1)	*l.d f6,-8(r1)*			
l.d f10,-16(r1)	*l.d f14,-24(r1)*			
l.d f18,-32(r1)	*l.d f22,-40(r1)*	*add.d f4,f0,f2*	*add.d f8,f6,f2*	
l.d f26,-48(r1)		*add.d f12,f10,f2*	*add.d f16,f14,f2*	
		add.d f20,f18,f2	*add.d f24,f22,f2*	
s.d f4, 0(r1)	*s.d f8,-8(r1)*	*add.d f28,f26,f2*		
s.d f12,-16(r1)	*s.d f16,-24(r1)*			*daddui r1,r1,#-56*
s.d f20, 24(r1)	*s.d f24,16(r1)*			
s.d f28,8(r1)				*bne r1,r2,loop*

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

MIPS superescalar..

- **Emisión de dos instrucciones por ciclo**
  - Una de operación de **punto flotante**, la otra "**entera**" (*load, store, branch*, operación entera)
    - ✓ En los procesadores actuales cualquiera de las dos puede ser la primera en el código secuencial
  - **Más sencillo** que permitir la emisión de dos instrucciones de cualquier tipo
- Fetch: 64 bits en cada ciclo de reloj

Ejecución sin paradas en MIPS superescalar..

- Simplificación: la instrucciones FP permanecen tres ciclos en la etapa EX (X)

Tipo instrucción	*Etapas*
Entera	**F D X M W**
Punto flotante	**F D X X X W**
Punto flotante	**F D X X X W**
Entera	**F D X M W**
Punto flotante	**F D X X X W**
Entera	**F D X M W**

MIPS superescalar..

- MIPS superescalar **requiere** una **considerable cantidad de operaciones FP** en los programas para que la mejora de rendimiento sobre el MIPS escalar sea sustancial
- 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

MIPS 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

MIPS superescalar..

- Puede aparecer una **dependencia RAW** entre las dos instrucciones del par (*load* FP y ALU FP o ALU FP y *store* FP)
  - En ese caso sólo se emite la primera
  - 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**

Bucle (de pág. 6) planificado y desenrollado para MIPS superescalar..

	Instrucción entera	*Instrucción FP*	*Ciclo reloj*
Loop:	**L.D F0, 0 (R1)**		**1**
	L.D F6, -8 (R1)		**2**
	L.D F10, -16 (R1)	**ADD.D F4, F0, F2**	**3**
	L.D F14, -24 (R1)	**ADD.D F8, F6, F2**	**4**
	L.D F18, -32 (R1)	**ADD.D F12, F10, F2**	**5**
	S.D F4, 0 (R1)	**ADD.D F16, F14, F2**	**6**
	S.D F8, –8 (R1)	**ADD.D F20, F18, F2**	**7**
	S.D F12, –16 (R1)		**8**
	DADDUI R1, R1, # -40		**9**
	S.D F16, 16 (R1)		**10**
	BNE R1, R2, Loop		**11**
	S.D F20, 8 (R1)		**12**

MIPS superescalar..

- Obsérvense las dependencias entre el primer L.D y el primer ADD.D y entre éste y el primer S.D
  - Ello **exige** planificar el bucle y **desenrollarlo cuatro veces** para evitar paradas de la unidad entera
  - 12ciclos/5 = **2,4 ciclos** por elemento del vector
    - ✓Recordemos: 14 ciclos / 4 = 3,5 ciclos por elemento, en MIPS escalar (planificando y desenrollando el bucle)
    - ✓La **mejora** ha sido de **1,5 veces**

Comparación VLIW/superescalares..

- 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 VLIW/superescalares..

- VLIW: el compilador depende de las características tecnológicas del procesador
  - Para una planificación eficiente, el compilador debe tener en cuenta detalles tecnológicos (por ejemplo, latencias de las 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 VLIW/superescalares

- 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 descodificarla
  - 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 VLIW y superescalares..

- (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 VLIW y superescalares

- 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