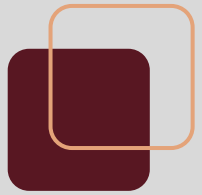
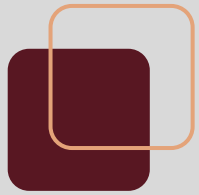
Aspectos avanzados de arquitectura de computadoras Vector Processors & Multithreading



Facultad de Ingeniería - Universidad de la República

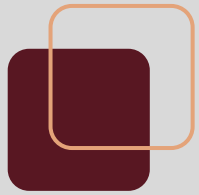
Curso 2018

Introducción



- En este capítulo se explorarán otras técnicas de aceleración de programas secuenciales:
- *Vector Processors*: Procesadores con registros vectoriales.
- *Multithreading:* La posibilidad de ejecutar múltiples *hilos (threads)* de forma *concurrente* en un mismo CPU.

Motivación (1/3)



- A medida que se aumenta el ancho de un superescalar, aumenta el costo del procesador, la potencia requerida y por tanto también la energía disipada (en calor).
- ¿Cuánto es el ancho *óptimo* para un superescalar?
- ¿Se aprovecha todo el poder de cómputo?

Motivación (2/3)

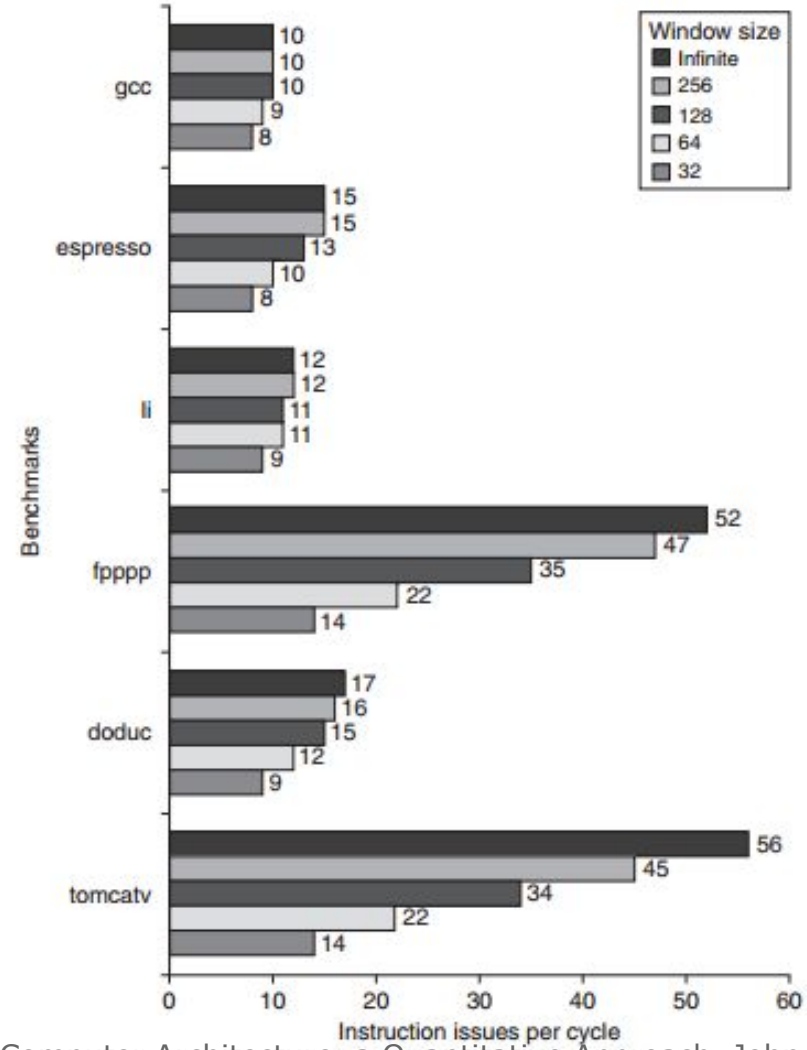
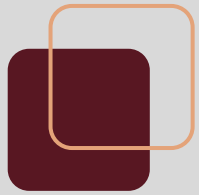


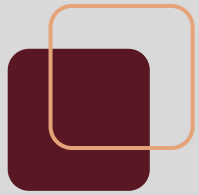
Imagen tomada del libro Computer Architecture, a Quantitative Approach. John L. Hennessy y David Patterson (2011)

Motivación (3/3)



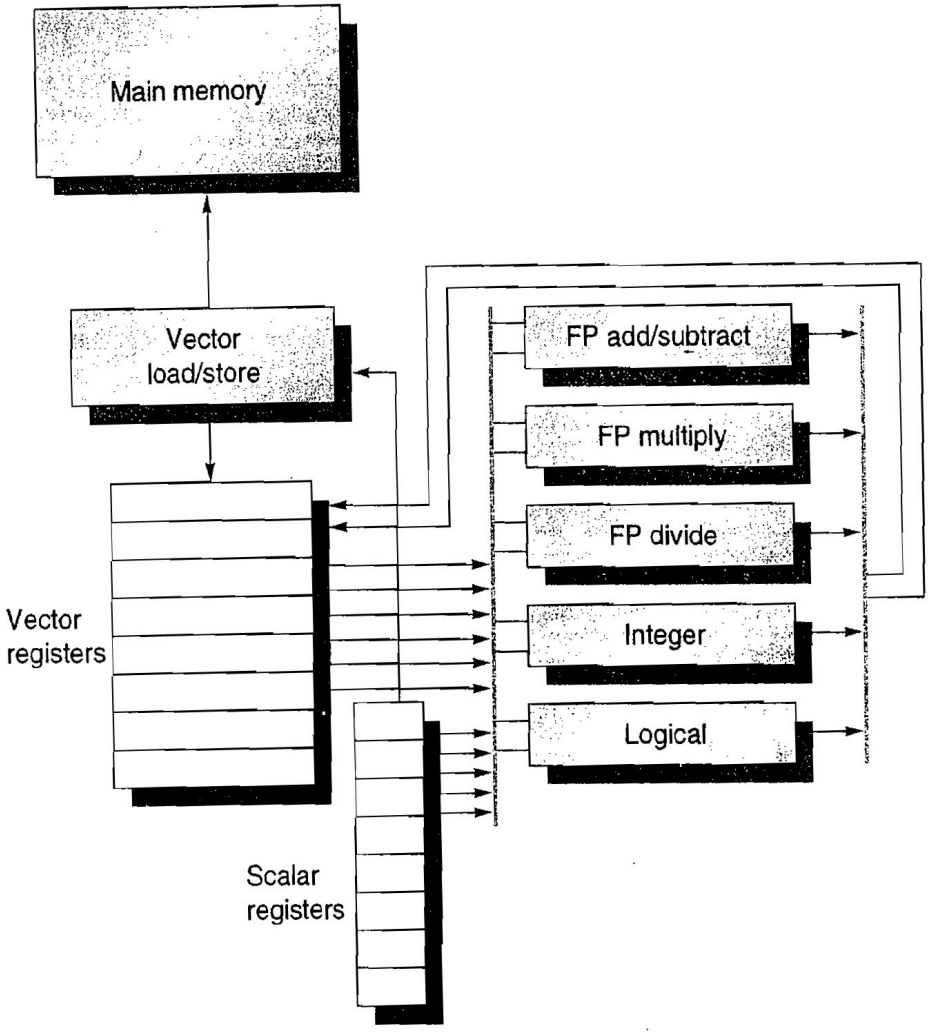
- Dificultad en exponer suficiente ILP conduce a nuevas formas para generar paralelismo.
- Una solución es contar con ayuda explícita del programador (esquemas VLIW, procesadores vectoriales).
- Algunos programas son naturalmente divisibles en hilos (*threads*). Diremos que esos programas exhiben *thread-level parallelism (TLP).*

Implementación VMIPS (1/2)

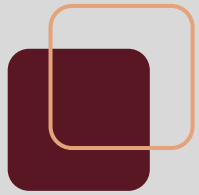


- Registros de vectores
  - 64 elementos de 8 bytes c/u por registro (512 bytes).
- Unidades funcionales vectoriales
- Load/Store vectoriales
- Registros escalares

Implementación VMIPS (2/2)



VMIPS ISA (1/2)



- ADDVV.D V1, V2, V3

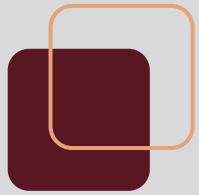
  Suma V2 y V3 y guarda en V1
- ADDVS.D V1, V2, R1

  Suma R1 a cada elemento de V2 y guarda en V1
- LV V1, R1

  Carga el Vector V1 desde memoria comenzando en la dirección R1
- SV R1, V1

  Guarda el Vector V1 desde memoria comenzando en la dirección R1

VMIPS ISA (2/2)



- LVWS V1, (R1, R2)

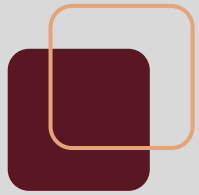
  Carga V1 desde memoria, comenzando en R1 y con desplazamiento R2 (V1i = R1 + i * R2)

- SVWS (R1, R2), V1
- MTC1 VLR, R1

  Guardar R1 en el *Vector Length Register*

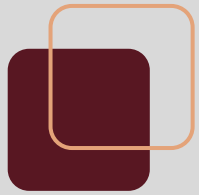
- MFC1 R1, VLR

Escalar vs Vectorial (1/2)



```
        L.D     F0,a          ;load scalar a
        DADDIU  R4,Rx,#512    ;last address to load
Loop:   L.D     F2,0(Rx)      ;load X[i]
        MUL.D   F2,F2,F0      ;a × X[i]
        L.D     F4,0(Ry)      ;load Y[i]
        ADD.D   F4,F4,F2      ;a × X[i] + Y[i]
        S.D     F4,9(Ry)      ;store into Y[i]
        DADDIU  Rx,Rx,#8      ;increment index to X
        DADDIU  Ry,Ry,#8      ;increment index to Y
        DSUBU   R20,R4,Rx     ;compute bound
        BNEZ    R20,Loop      ;check if done
```

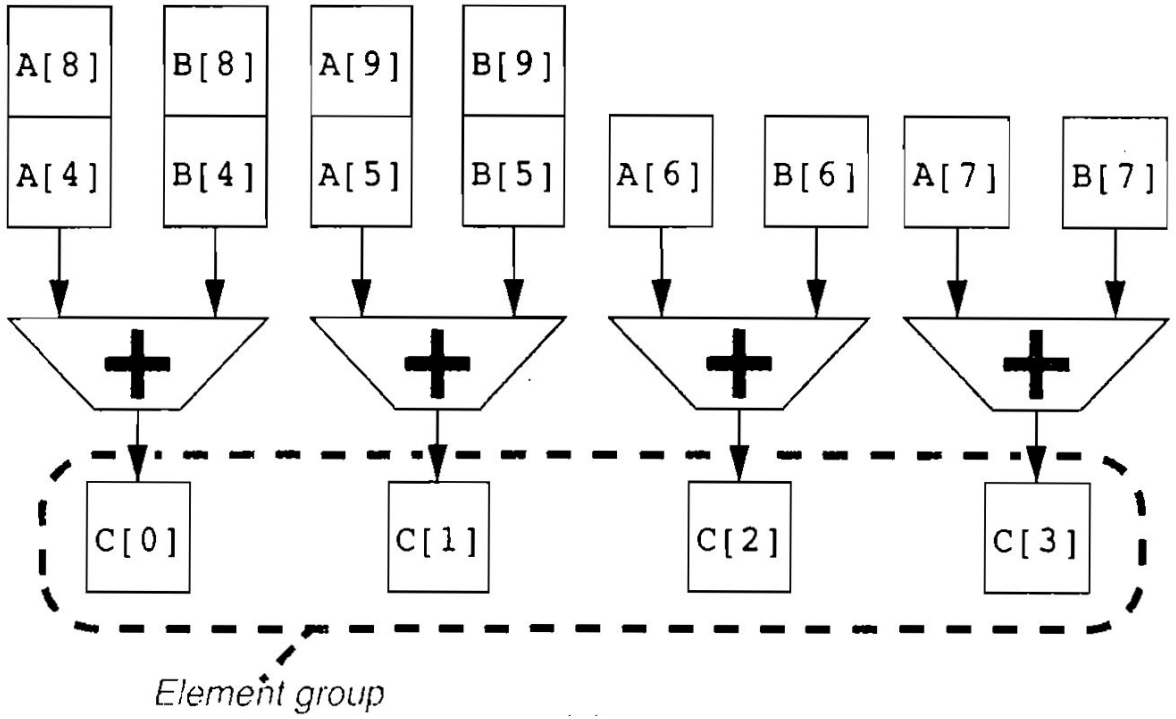
Escalar vs Vectorial (2/2)



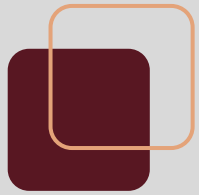
```
L.D       F0,a       ;load scalar a
LV        V1,Rx      ;load vector X
MULVS.D   V2,V1,F0   ;vector-scalar multiply
LV        V3,Ry      ;load vector Y
ADDVV.D   V4,V2,V3   ;add
SV        V4,Ry      ;store the result
```

Ejecución Vectorial

- *Chaining*
- Múltiple lanes

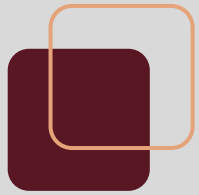


Vectorial vs Superescalar OoO



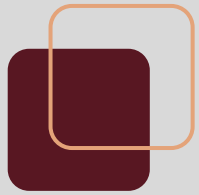
- Tradeoff:
  - Costo
  - Consumo
  - Lógica de control
  - Flexibilidad
- Usos:
  - Aplicaciones científicas
  - Multimedia

Implementaciones Reales



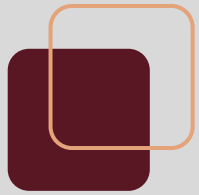
- Cray 1 (1976)
- Cray 2 (1985)
- Cray C90 (1991)
- Cray T90 (1995)
- VMIPS (2001)
- Cray X1 (2002)
- Cray XIIE (2005)
- Línea Intel Xeon Phi

Extensiones multimedia (1/4)



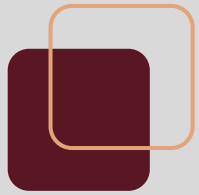
- Extensiones a ISAs para manejar instrucciones *tipo vectoriales* (SiMD)
- x86:
  - MMX (1997)
  - SSE (1999)
  - AVX (2011)

Extensiones multimedia (2/4)



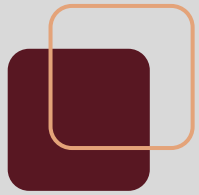
- MMX:
  - Permite utilizar registros de 64 bits como 2 registros de 32 bits, 4 de 16 u 8 de 8 bits.
- SSE:
  - 8 nuevos registros de 128 bits de uso exclusivo de la extensión.
  - Primera implementación: Pentium III

Extensiones multimedia (3/4)



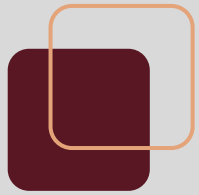
- Instrucciones SSE:
  - ADDPS, SUBPS, CMPPS, ANDPS, ORPS, XORPS, ANDPS, etc.
  - Operan dividiendo MMX en 4 subregistros de 32 bits.
  - SSE2 (2001), extiende para soportar 8 ops de 16 bits, 16 ops de 8 bits, etc.

Extensiones multimedia (4/4)



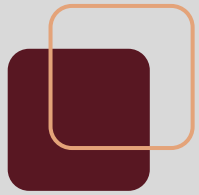
- AVX:
  - Extensión de registros MMX a 256 bits (renombrados a YMM).
- Nuevas instrucciones:
  - VBROADCASTSS
    - Copia un escalar en todas las posiciones de un vector
  - Adición de la varias de las instrucciones disponibles en VMIPS! (LVWS, Gather/Scatter, operaciones condicionales)

Multithreading (1/4)



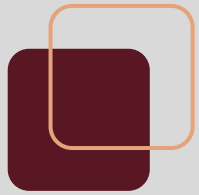
- *Multithreading:* La posibilidad de ejecutar múltiples *hilos (threads)* de forma *concurrente* en un mismo CPU.
- Los hilos pueden provenir de la multiprogramación (programas independientes) o de aplicaciones multihilo (hilos paralelos de un mismo programa).

Multithreading (2/4)



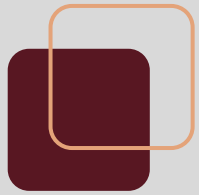
- Recursos duplicados:
  - Set de registros
  - PC
  - Tabla de páginas
- Recursos compartidos:
  - Todo lo demás! En particular unidades funcionales, cachés, predictores, etc.

Multithreading (3/4)



- Tipos de multithreading:
  - *Fine-Grained*: en cada ciclo de reloj se despacha una instrucción de un hilo diferente.
  - *Coarse-Grained*: ejecuta un thread por varios ciclos o hasta que ocurre una detención 'larga'.

Multithreading (4/4)



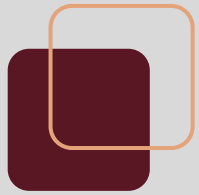
- Tipos de multithreading:
  - *Simultaneous Multithreading (SMT)*: Se aplica el nombre cuando se utiliza *fine-grained* sobre un procesador superescalar. Se despachan varias instrucciones por segundo, las cuales pueden ser de uno o más threads, según disponibilidad.

Fine-Grained Multithreading (1/2)

- Fine-grained multithreading en el pipeline de 5 etapas, con 4 threads diferentes en ejecución:

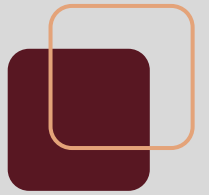
Instrucción\Ciclo	1	2	3	4	5	6	7	8	9	10
thread 1	IF	ID	EX	MEM	WB					
thread 2		IF	ID	EX	MEM	WB				
thread 3			IF	ID	EX	MEM	WB			
thread 4				IF	ID	EX	MEM	WB		
thread 1					IF	ID	EX	MEM	WB	
thread 2						IF	ID	EX	MEM	WB

Fine-Grained Multithreading (2/2)



- Notar que cuando la segunda instrucción del hilo comienza, la primera ya escribió sus resultados
  - No hay dependencias de datos ni de control!
- Latencia de cada thread mayor que si fuera el único en ejecución.

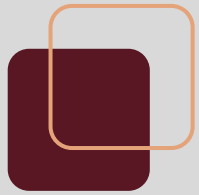
Coarse-grained Multithreading (1/2)



- Coarse-grained multithreading en el pipeline de 5 etapas, con 2 threads diferentes en ejecución y granularidad de 3 etapas:

Instrucción\Ciclo	1	2	3	4	5	6	7	8	9	10
thread 1	IF	ID	EX	MEM	WB					
thread 1		IF	ID	EX	MEM	WB				
thread 1			IF	ID	EX	MEM	WB			
thread 2				IF	ID	EX	MEM	WB		
thread 2					IF	ID	EX	MEM	WB	
thread 2						IF	ID	EX	MEM	WB

Coarse-grained Multithreading (2/2)



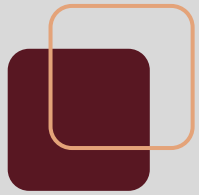
- El esquema de coarse-grained consiste en una ejecución de granularidad mayor, cambiando de thread cada mayor cantidad de ciclos que en fine-grained, o cuando un thread se detiene por una latencia *larga*
  - Ejemplo: L2 cache miss
- Permite la 'ilusión' de ejecutar un solo programa, sin perder poder de cómputo en detenciones largas

Simultaneous Multithreading (1/4)

- Simultaneous multithreading en un superescalar OoO con dos despachos por ciclo y un pipeline de 5 etapas:

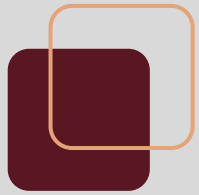
Instrucción\Ciclo	1	2	3	4	5	6	7	8
thread 1	IF	ID	EX	MEM	WB			
thread 2	IF	ID	EX	MEM	WB			
thread 3		IF	ID	EX	MEM	WB		
thread 4		IF	ID	EX	MEM	WB		
thread 4			IF	ID	EX	MEM	WB	
thread 4			IF	ID	EX	MEM	WB	
thread 1				IF	ID	EX	MEM	WB
thread 3				IF	ID	EX	MEM	WB

Simultaneous Multithreading (2/4)



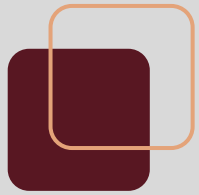
- SMT es "*fácilmente*" incorporable a un procesador OoO con despacho dinámico porque pueden compartirse muchos recursos de hardware:
  - Ventana de instrucciones
  - Registros de renombre
  - Unidades funcionales

Simultaneous Multithreading (3/4)



- Políticas de elección de hilo:
  - Fixed interleave: tipo fine-grained multithreading, elegir instrucciones en formato round-robin hasta el issue-width de la máquina
  - Prioridad: definir prioridades en tiempo de carga de threads.

Simultaneous Multithreading (4/4)



- Otras estrategias:
  - Elegir instrucciones ‘complejas’ primero (saltos condicionales, accesos a memoria).
  - Ejecutar instrucciones del hilo con mayor cantidad de instrucciones en la ventana de instrucciones.
  - Fetch de threads con pocas instrucciones en la ventana.

Multithreading

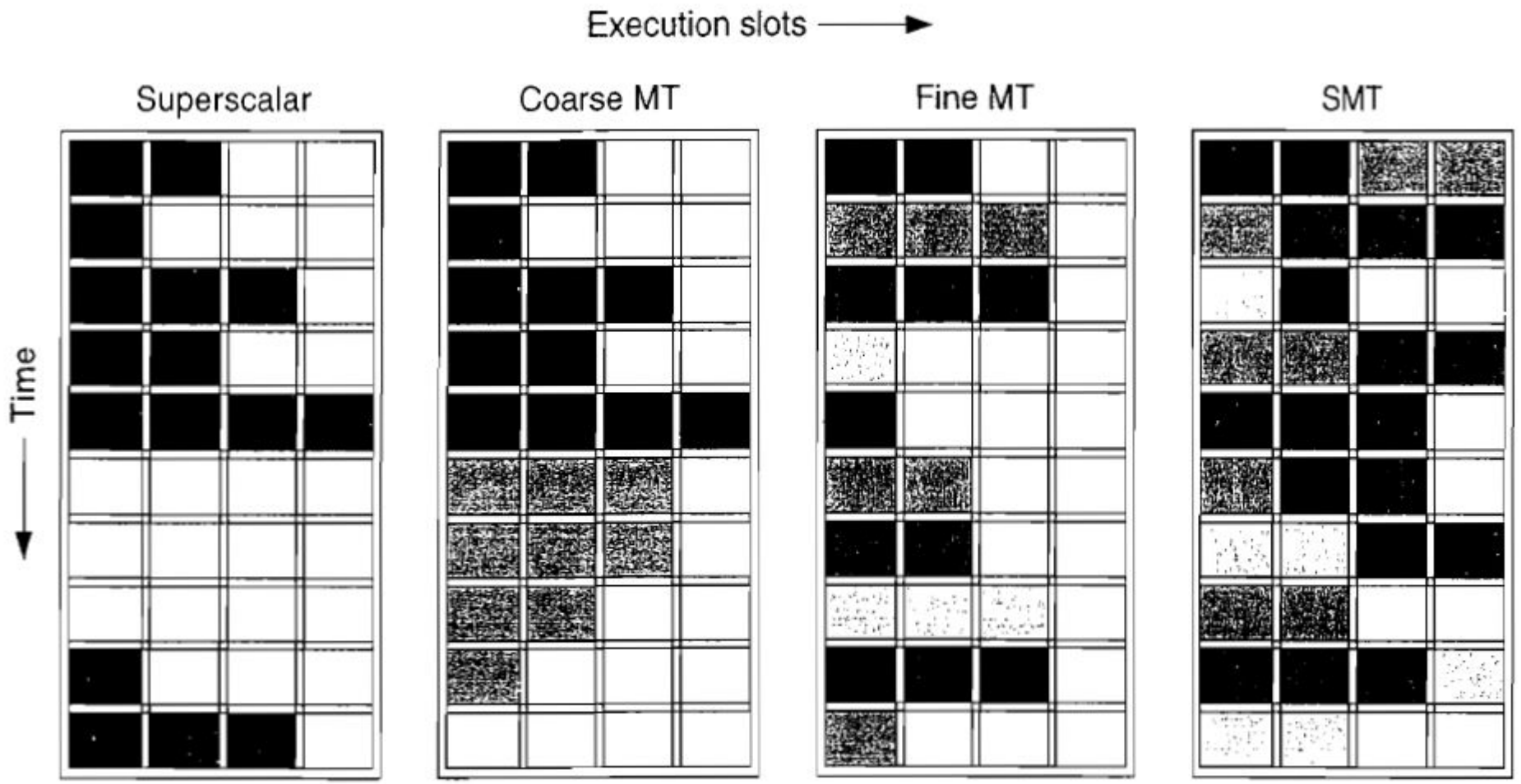
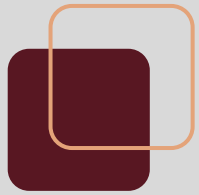


Imagen tomada del libro Computer Architecture, a Quantitative Approach. John L. Hennessy y David Patterson (2011)

Multithreading



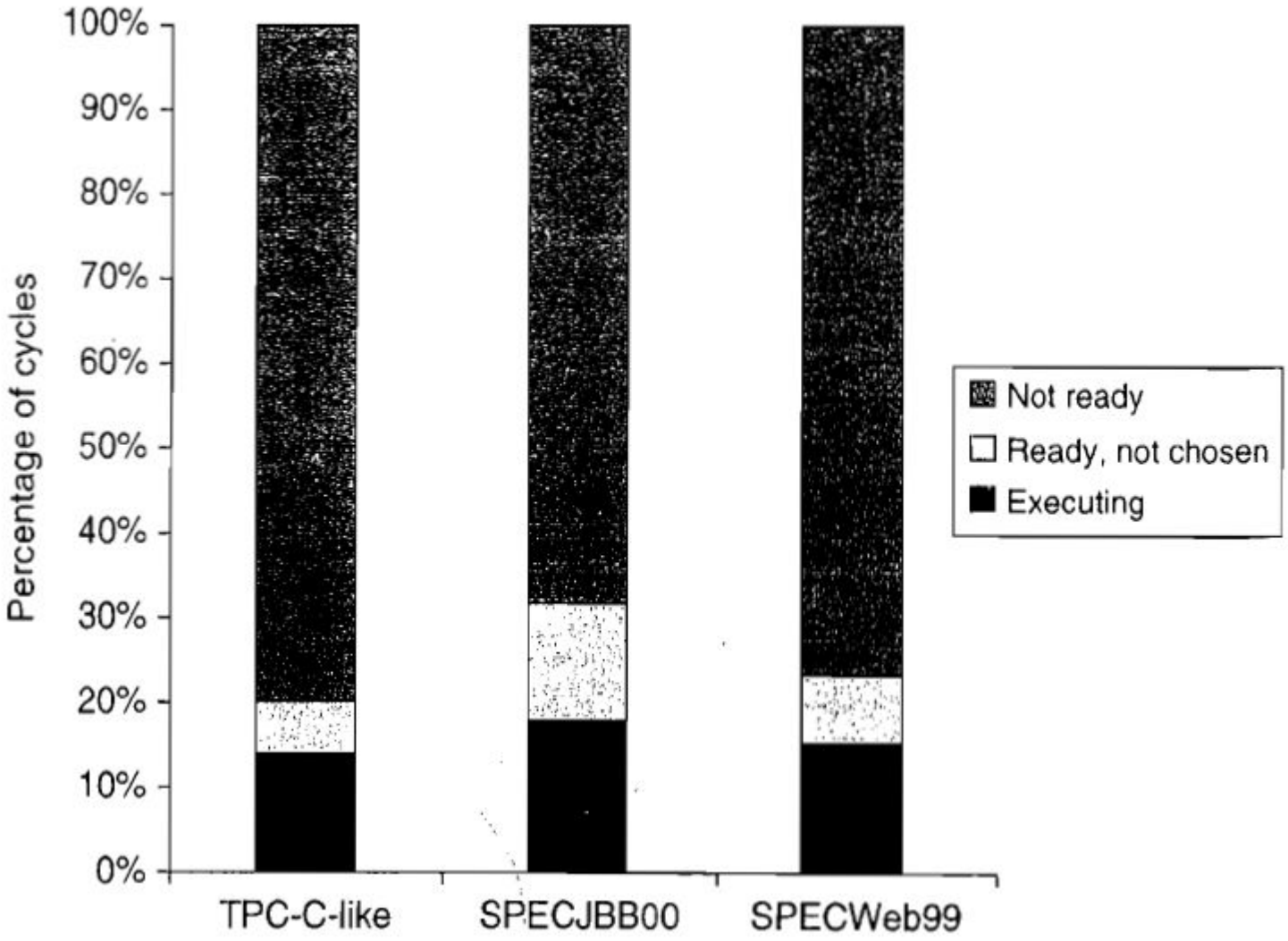
- Los cpu’s modernos utilizan casi que universalmente SMT
  - Sun T1 y T2 (Fine grained - 8 threads) - 2005 y 2007
  - IBM Power 7 (SMT - 4 threads) - 2010
  - Intel core i7 (SMT - 2 threads) - “Intel Hyperthreading”

Sun T1

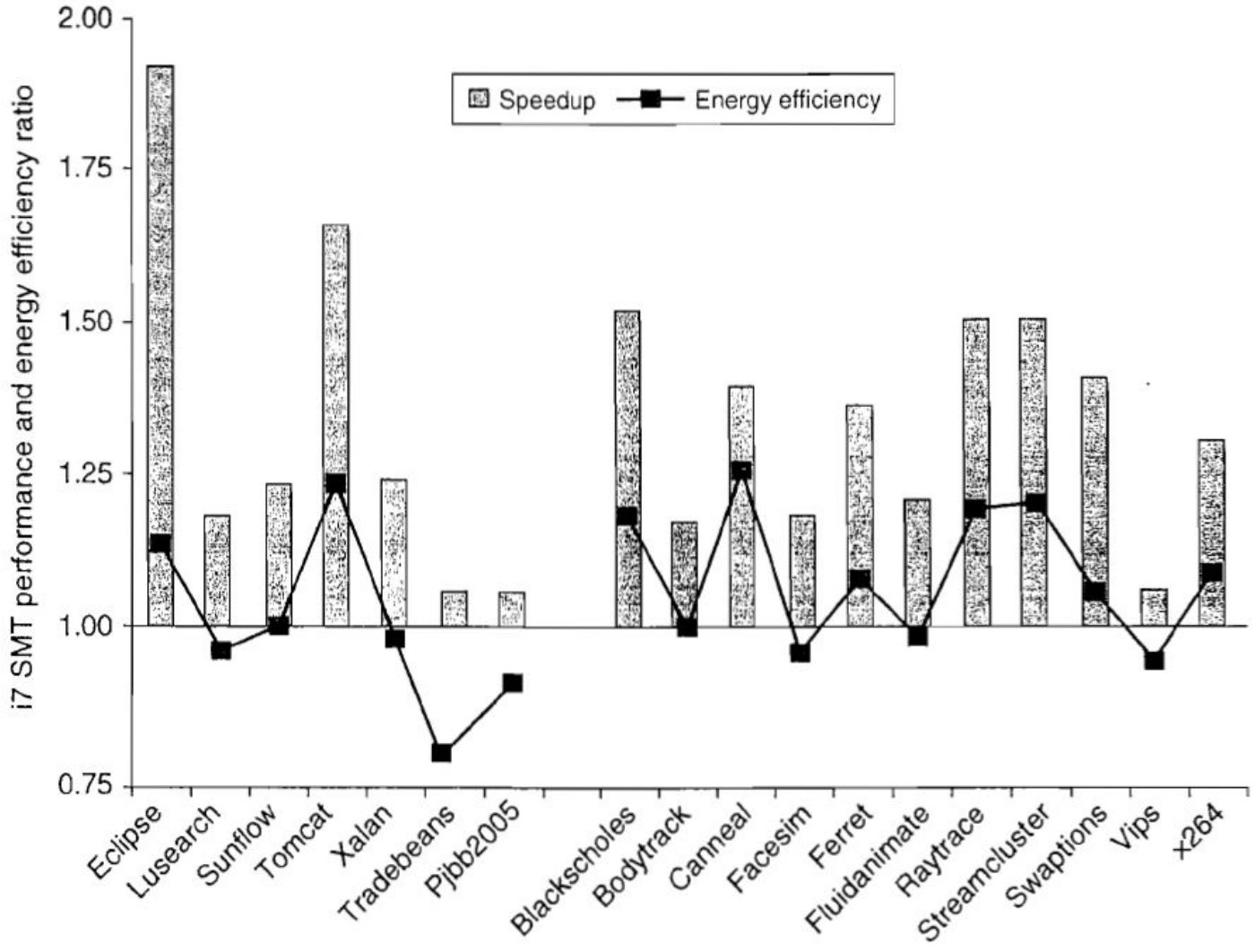
Characteristic	Sun T1
Multiprocessor and multithreading support	Eight cores per chip; four threads per core. Fine-grained thread scheduling. One shared floating-point unit for eight cores. Supports only on-chip multiprocessing.
Pipeline structure	Simple, in-order, six-deep pipeline with three-cycle delays for loads and branches.
L1 caches	16 KB instructions; 8 KB data. 64-byte block size. Miss to L2 is 23 cycles, assuming no contention.
L2 caches	Four separate L2 caches, each 750 KB and associated with a memory bank. 64-byte block size. Miss to main memory is 110 clock cycles assuming no contention.
Initial implementation	90 nm process; maximum clock rate of 1.2 GHz; power 79 W; 300 M transistors; 379 mm^2 die.

Imagen tomada del libro Computer Architecture, a Quantitative Approach. John L. Hennessy y David Patterson (2011)

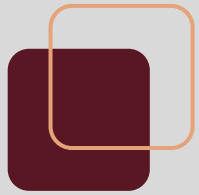
Análisis sobre T1



Performance SMT en i7



Fin



¿Preguntas?