

Arquitectura de Computadores

Grado de Informática

Simulación

Departament d'Enginyeria Informàtica I Matemàtiques

Universitat Rovira i Virgili

Tarragona, Spain



Índice

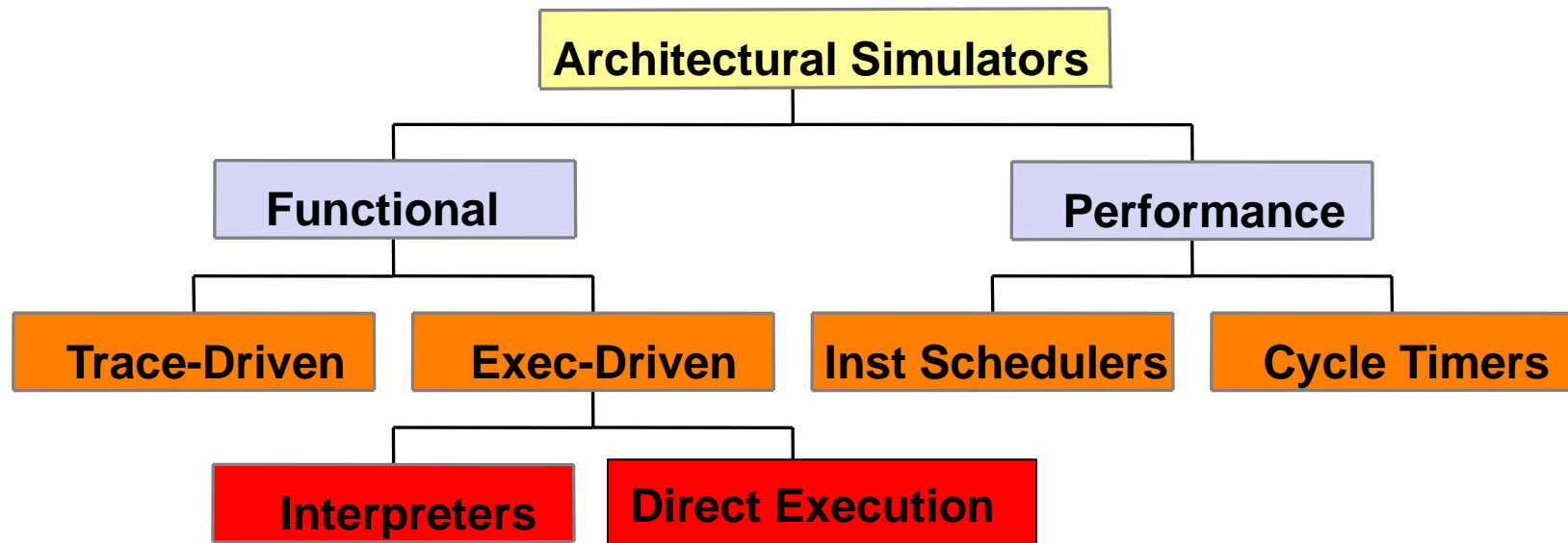
I. Simuladores

II. Simplexscalar

III. Benchmarks

Simuladores

■ Clasificación de Simuladores



Functional vs Performance

■ Functional Simulators

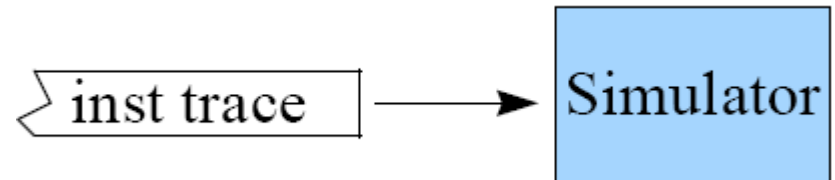
- Implementan la arquitectura (ISA)
- Realizan una ejecución real
- Implementan lo que el programador ve

■ Performance Simulators

- Implementan la microarquitectura (implementación del ISA)
- Modelan los recursos del sistema
- Preocupados por el tiempo
- Implementan lo que el programador no ve
- También llamados timing simulators

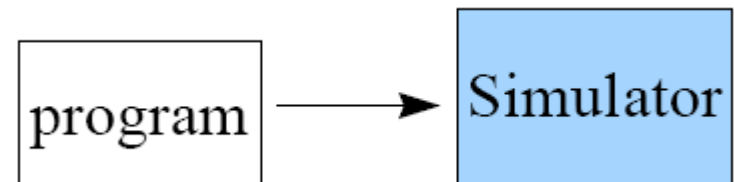
Trace-Driven vs Execution Driven

■ Trace-Driven



- Simulador lee traza de instrucciones captada en ejecución previa
- Fácil de implementar
- No hay información, ni evaluación de ejecución especulativa
- No se modela un comportamiento real del todo

■ Execution-Driven



- Simulador ejecuta el programa. No hay ejecución previa
- Difícil de implementar
- Permite la evaluación de la ejecución especulativa
- Se valida un comportamiento real

Schedulers vs Cycle Timers

■ Instruction Schedulers

- Simulador lanza instrucciones cuando recursos están disponibles
- Las instrucciones se procesan una a una
- Simple pero menos detallado

■ Cycle Timers

- Simulador sigue el estado de la microarquitectura ciclo a ciclo
- Estado del simulador = estado de la microarquitectura
- Perfecto para la simulación de la microarquitectura

Índice

I. Simuladores

II. Simplescalar

III. Benchmarks

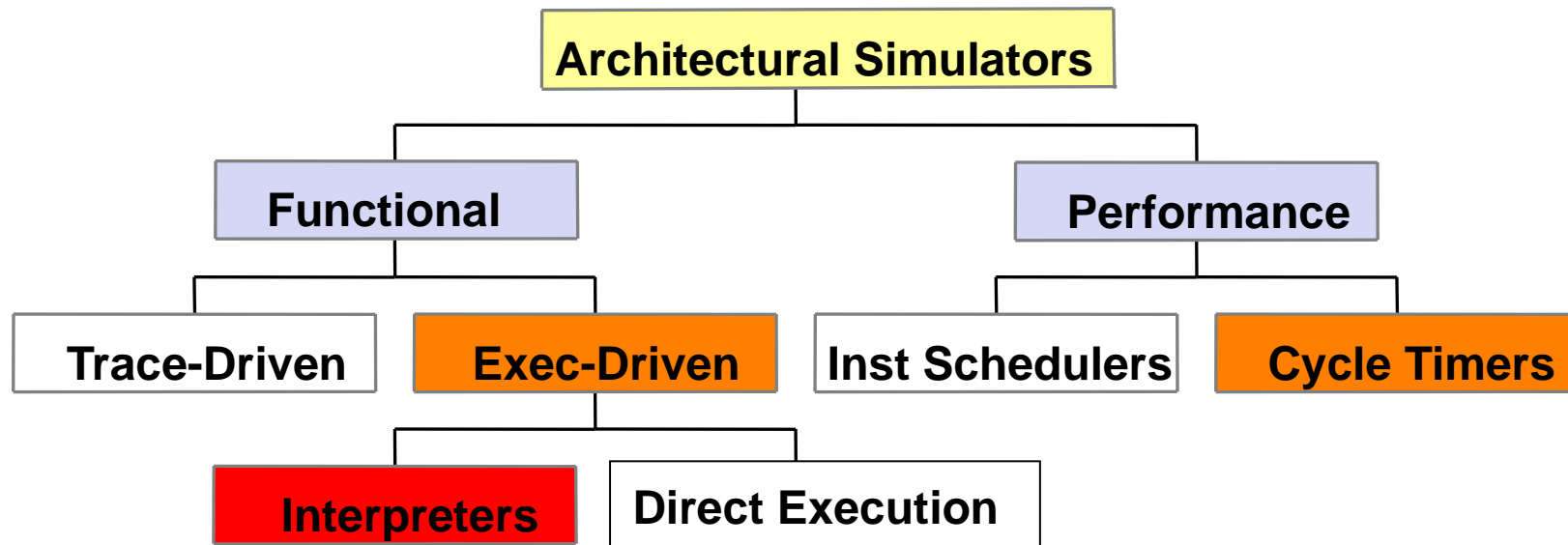
Visión General

Introducción



- **Conjunto de herramientas para la simulación**
 - desde simples procesadores moniclo
 - a complejos procesadores superescalares
 - ejecución fuera de orden
 - jerarquía de memoria completa
- **Universidad de Wisconsin-Madison**
- **Licencia libre y código abierto para fines académicos**
- **Ampliamente usado por la comunidad investigadora**
- **Limitaciones:**
 - Single core
 - No proporciona datos sobre consumo

Tipo de Simulador

- En color las opciones incluidas en SimpleScalar



Simplescalar Suite

Sim-Fast	Sim-Safe	Sim-Profile	Sim-Cache Sim-BPred	Sim-Outorder
300 lines	350 lines	900 lines	< 1000 lines	3900 lines
functional	functional	functional	functional	performance
No timing	w/checks	Lot of stats	Cache stats Branch stats	OoO issue Branch pred. Mis-spec. ALUs Cache TLB 200+ KIPS
+ Velocidad - 				
- Precisión y Detalle + 				

Sim-Fast, Sim-Safe

■ Sim-Fast

- Simulador funcional
- Optimizado en velocidad
- No asume jerarquía de memoria
- No valida las instrucciones

■ Sim-Safe

- Simulador funcional
- Optimizado en velocidad
- No asume jerarquía de memoria
- Valida las instrucciones

Sim-Profile, Sim-BPred

■ Sim-Profile

- Más que simulador es una herramienta de profiling
- Permite la recolección de información de profiling
- Genera informes detallados sobre:
 - Tipos de instrucción
 - Tipos de saltos
 - Tipos de accesos a memoria
 - etc

■ Sim-Bpred

- Simulador centrado en parte específica del procesador
- Analiza predictores de saltos
 - taken, not taken, perfect, bimodal, 2level, hybrid
- Genera estadísticas de aciertos y fallos del predictor
- No se reflejan los efectos del predictor en el tiempo de ejecución

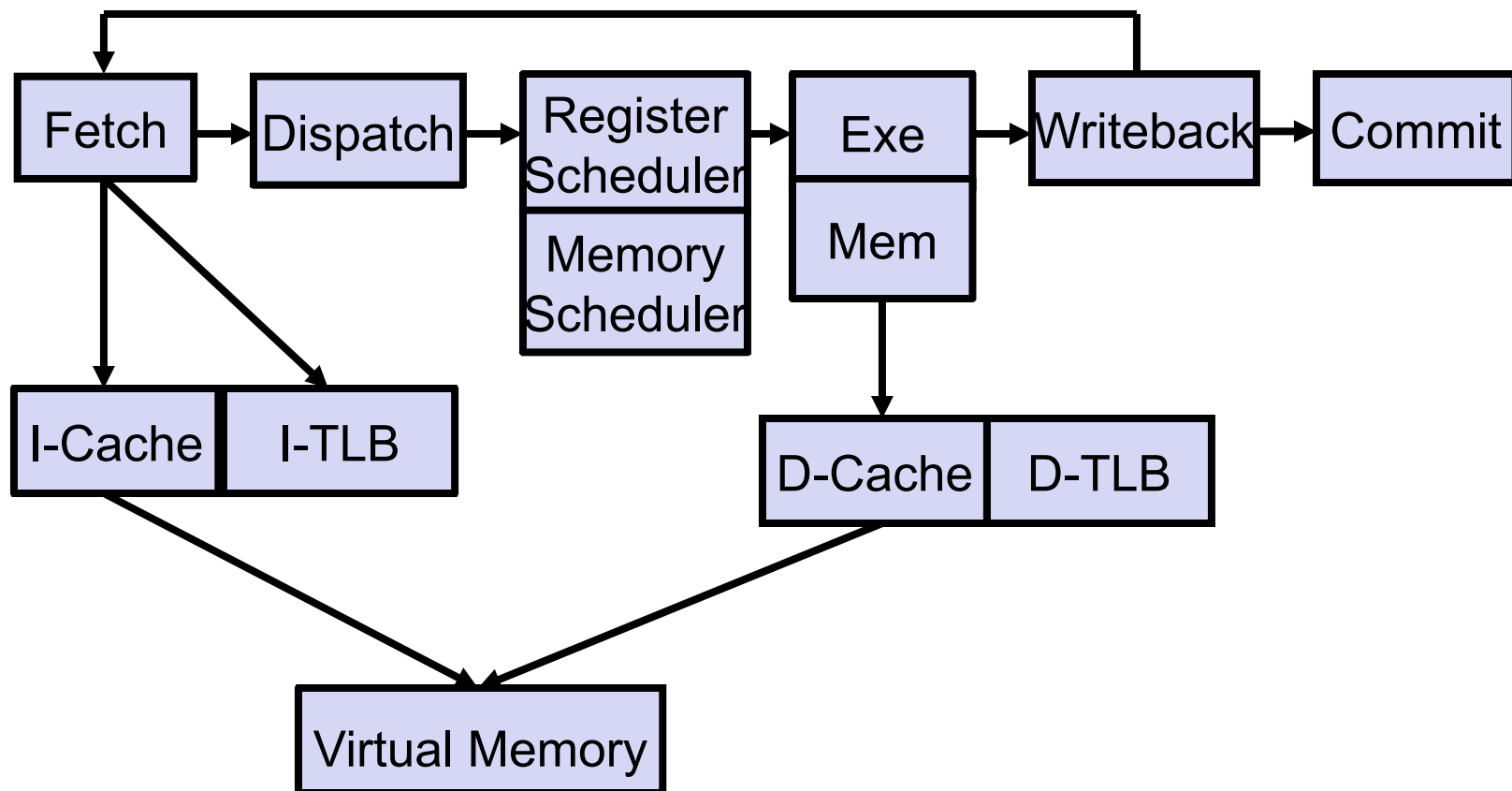
Sim-Cache

- **Simulador específico y rápido**
- **Centrado en la jerarquía de memoria del procesador**
- **Genera estadísticas de aciertos y fallos en jerarquía**
- **No refleja efectos de memoria en tiempo de ejecución**
- **Permite simular:**
 - 2 niveles de caché
 - separar (o no) datos de instrucciones
 - TLB
 - Vaciado (flush) y compresión
- **Ideal si sólo nos centramos en memoria**

Sim-Outorder

- **El simulador más complicado y detallista**
- **Intenta simular todas las partes de un procesador**
 - Jerarquía de memoria
 - Predictor de saltos
 - Ejecución fuera de orden
 - Banco de registros
 - Estaciones de reserva
 - Buffers de memoria
- **Permite parametrizar la mayoría de sus componentes**
- **Se puede modificar el código para extender/añadir nuevas funcionalidades**

Sim-Outorder (Hardware)



Instalación

Pasos de Instalación

- **Obtener última versión de** `www.simplescalar.com`
 - `simplesim-3v0e.tgz`
- **Descomprimir**
 - `tar xvf simplesim-3v0e.tgz`
- **Configurar uno de dos ISA disponibles**
 - `make config-pisa`
 - `make config-alpha`
- **Compilar todas las herramientas**
 - `make`
- **Ahora ya se pueden realizar simulaciones**
 - `sim-outorder`, `sim-cache`, `sim-fast`, etc

Ejecución y Configuración

Ejecutar Simulador

- **El formato para ejecución por línea de comandos es:**

```
simulador {-parametros} benchmark  
{argumentos} >& output_file_name
```

- **Simulador**

- sim-cache, sim-outorder, etc

- **Formato de los parámetros de simulación:**

```
-parametroX valor
```

- Se ponen por línea de comandos uno detrás de otro

- **Benchmarks**

- spec2006, minibench, etc

Parámetros Generales

- **-config nom_fichero**
 - permite poner todos los parámetros en un fichero
 - en el fichero cada parámetro va en una línea diferente
- **-fastfwd valor**
 - número de instrucciones iniciales saltadas sin simular
 - las instrucciones se ejecutan pero no recolectan estadísticas
 - sólo disponible en sim-outorder
- **-max:inst valor**
 - máximo número de instrucciones ejecutadas y simuladas
- **-redir:sim nom_fichero**
 - redirecciona salida del simulador en un fichero
- **-redir:prog nom_fichero**
 - redirecciona salida del benchmark en un fichero

Parámetros del Superescalar

Parámetro	Argumento	Valor por Defecto
-fetch:ifqsize -decode:width -issue:width -commit:width	<int>	4
Número de instrucciones que se tratan a la vez por ciclo en cada etapa		
-ruu:size	<int>	16
Número de instrucciones que es capaz de almacenar el procesador para la ejecución simultanea “fuera de orden”		
-lsq:size	<int>	8
Número de instrucciones de acceso a memoria simultaneas		

Parámetros de Memoria

Parámetro	Argumento	Valor por Defecto
-mem:lat	<int> <int>	18 2
Latencia en ciclos de acceso a memoria de un bloque de cache. Para el primer acceso del bloque Para los siguientes accesos hasta completar el bloque		
-mem:width	<int>	8
Tamaño en bytes del bus de acceso a memoria		

Si un bloque de cache son 32 bytes:

Suponen 4 accesos a memoria con una

Latencia de $18 + 2 + 2 + 2$ ciclos

Configuración del superscalar

Si queremos un superescalar de 8 vías,
con una ventana de instrucciones de 256
y una cola de accesos a memoria de 32

■ Configuración:

-fetch:ifqsize	8
-decode:width	8
-issue:width	8
-commit:width	8
-ruu:size	256
-lsq:size	32

Configuración del acceso a memoria

Bus de acceso a memoria de 32 bytes con
una latencia de 300 y 2 ciclos

■ Configuración:

-mem:lat	300	2
-mem:width	32	

Estadísticas Resultados

Estadísticas Simulador (Generales)

sim_num_insn	total number of instructions committed
sim_num_refs	total number of loads and stores committed
sim_num_loads	total number of loads committed
sim_num_stores	total number of stores committed
sim_num_branches	total number of branches committed
sim_elapsed_time	total simulation time (seconds)
sim_total_insn	total number of instructions executed
sim_total_refs	total number of loads and stores executed
sim_total_loads	total number of loads executed
sim_total_stores	total number of stores executed
sim_total_branches	total number of branches executed
sim_cycle	total simulation time (cycles)
sim_IPC	instructions per cycle
sim_CPI	cycles per instruction

Estadísticas Simulador Superescalar

■ Estadísticas de los buffers de instrucciones:

- IFQ → buffer de fetch de instrucciones

IFQ_count	Suma de las instrucciones que hay en cada ciclo
IFQ_fcount	Veces que el buffer esta lleno
ifq_occupancy	Media de instrucciones en el buffer por ciclo
ifq_rate	Media de instrucciones salen/entran por ciclo
ifq_latency	Media de ciclos que esta en el buffer
ifq_full	Media de veces que el buffer esta lleno

■ Estas estadísticas se repiten para:

- RUU → ventana de instrucciones
- LSQ → buffer de accesos a memoria en marcha

Índice

I. Simuladores

II. Simplexscalar

III. Benchmarks

Test Benchmarks

- **Simplescalar incorpora benchmarks de prueba**
 - `simplesim-3.0/tests-pisa/`
 - `test-math`, `test-printf`, etc
- **Pocas instrucciones, ejecución instantánea**
- **Sirven para validar el funcionamiento del simulador**
- **Ejemplos de ejecución**

`sim-cache simplesim-3.0/tests-pisa/bin.little/test-math`
- **Se puede probar simulador y todos los tests**

```
make sim-tests
```

Spec CPU2000 (Overview)

- **SPEC: System Performance Evaluation Cooperative**
- **Sociedad sin ánimo de lucro**
- **Misión:**
 - establecer, mantener y distribuir un conjunto estandarizado de *benchmarks* que se pueden aplicar a las últimas generaciones de procesadores
- **Spec CPU2000:**
 - enfocado a sistemas de computadores de carácter general
 - representativos del 2000 al 2006
- **Se dividen en dos grupos**
 - SpecINT: programas con cálculos de aritmética entera
 - SpecFP: programas con cálculos de aritmética en coma flotante

Spec CPU2000 (CINT)

Benchmark	Description
164.gzip	Compression
175.vpr	FPGA place and route
176.gcc	C compiler
181.mcf	Combinatorial optimization
186.crafty	Chess
197.parser	Word processing, grammatical analysis
252.eon	Visualization (ray tracing)
253.perlbmk	PERL script execution
254.gap	Group theory interpreter
255.vortex	Object-oriented database
256.bzip2	Compression
300.twolf	Place and route simulator

Spec CPU2000 (CFP)

Benchmark	Description
168.wupwise	Physics/Quantum Chromodynamics
171.swim	Shallow water modeling
172.mgrid	Multi-grid solver: 3D potential field
173.applu	Parabolic/elliptic PDE
177.mesa	3-D graphics library
178.galgel	Computational Fluid Dynamics
179.art	Image Recognition/Neural Networks
183.equake	Seismic Wave Propagation Simulation
187.facerec	Image processing: face recognition
188.amp	Computational chemistry
189.lucas	Number theory/primalty testing
191.fma3d	Finite-element Crash Simulation
200.sixtrack	High energy nuclear physics accelerator design
301.apsi	Meteorology: Pollutant distribution

Cargas de Trabajo (WorkLoads)

- **Cada benchmark dispone de 3 posibles “workloads”**
 - test
 - train
 - reference
- **TEST: ejecuta alrededor de 500 millones de Insts.**
- **TRAIN: ejecuta alrededor de 5000 millones de Insts.**
- **REFERENCE: ejecuta alrededor de 50.000 millones**
- **Lo ideal sería ejecutar todo el REFERENCE**
- **Habitualmente se coge un conjunto representativo**
 - Se saltan X millones (por ejemplo 200)
 - Se evalúan Y millones (por ejemplo 500)