AAAC 2016

Obligatorio 1
Simulador SimpleScalar

Principios del simulador

- ¿Qué es un simulador de arquitectura?
  - Herramienta que reproduce el comportamiento de un dispositivo de cómputo
- ¿Por qué usar un simulador?
  - Flexible
    - Diferentes diseños de hardware
    - Abstraerse todo lo necesario
  - Barato
- ¿Por qué no usar un simulador?
  - Lento
  - ¿Correctitud?

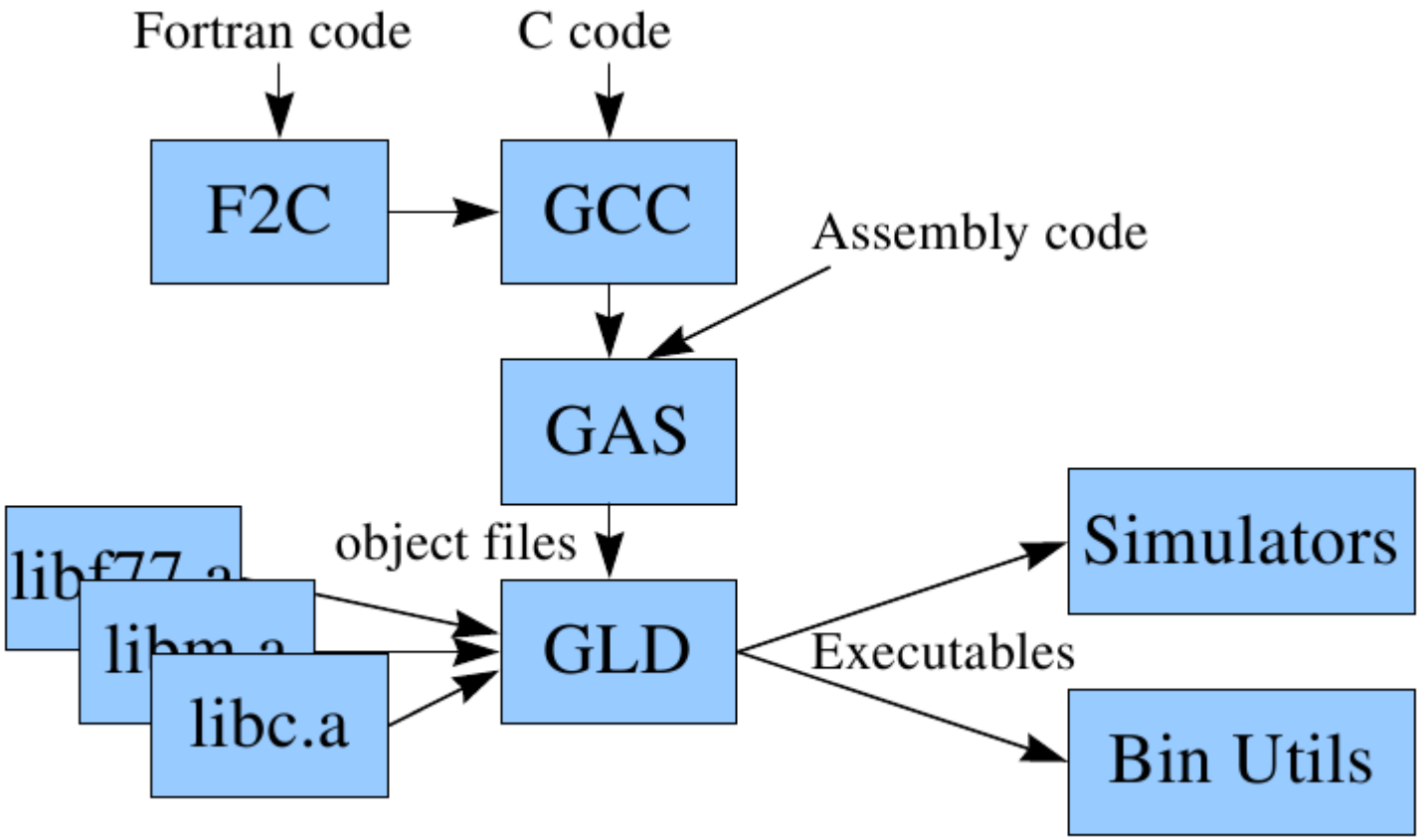
Funcionales vs. Rendimiento

- Simuladores funcionales implementan la arquitectura
  - Realizan la ejecución
  - Implementan lo que los programadores ven
- Simuladores de rendimiento implementan la microarquitecurta
  - Modelan los recursos del sistema
  - Miden el tiempo
  - Implementan lo que los programadores no ven.

Simplescalar

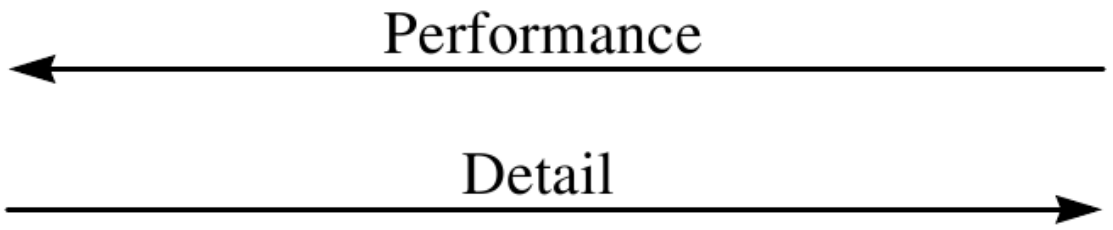
- Desarrollado por Todd Austin y Doug Burger en University of Wisconsin-Madison, ’94-’96
- Colección de simuladores que emulan un microprocesador a distintos niveles (funcional, funcional + cache/bpred, out-of-order, etc).
- El procesador simulado es un derivado de la arquitectura MIPS.
- Herramientas:
  - Compilador C, assembler, linker (para PISA)
  - DLite: debugger
  - Visor de trazas del pipeline

Simplescalar



Simplescalar

Sim-Fast	Sim-Safe	Sim-Profile	Sim-Cache/ Sim-Cheetah	Sim-Outorder
- 420 lines - functional - 4+ MIPS	- 350 lines - functional w/ checks	- 900 lines - functional - lot of stats	- < 1000 lines - functional - cache stats	- 3900 lines - performance - OoO issue - branch pred. - mis-spec. - ALUs - cache - TLB - 200+ KIPS



Sim-Bpred

- Se ejecuta el modelo detallado del predictor de saltos.
- Resultados rápidos para tasas de predicciones.
- Sin impacto de tiempo o rendimiento.

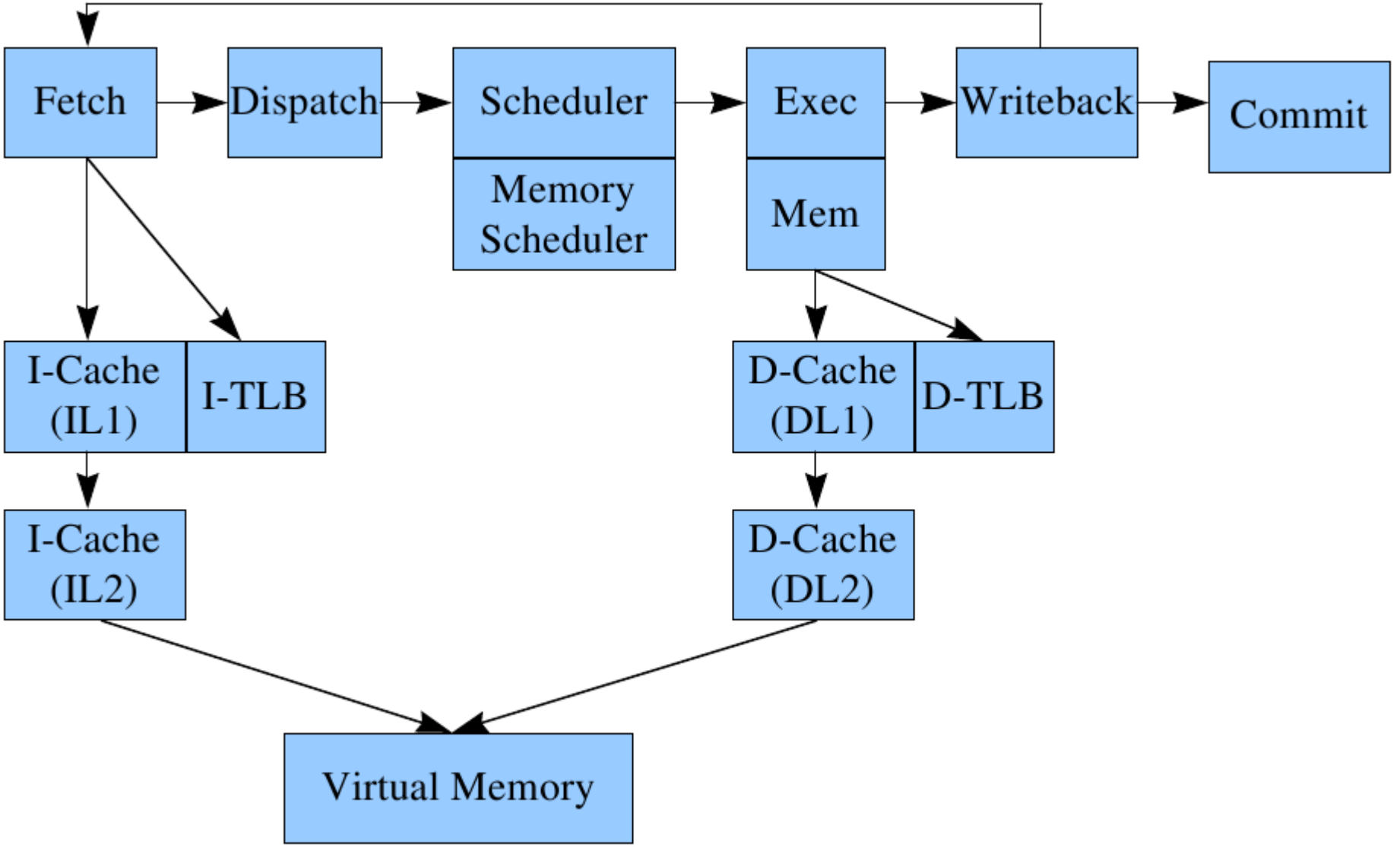
Predictores implementados

- Especificar tipo de predictor de saltos
  - -bpred <type>
- Predictores implementados
  - nottaken siempre predice que no se toma el salto
  - taken siempre predice que se toma el salto
  - perfect
  - bimod predictor bimodal (BHT con entradas de 2 bits)
  - 2lev predictor de 2 niveles

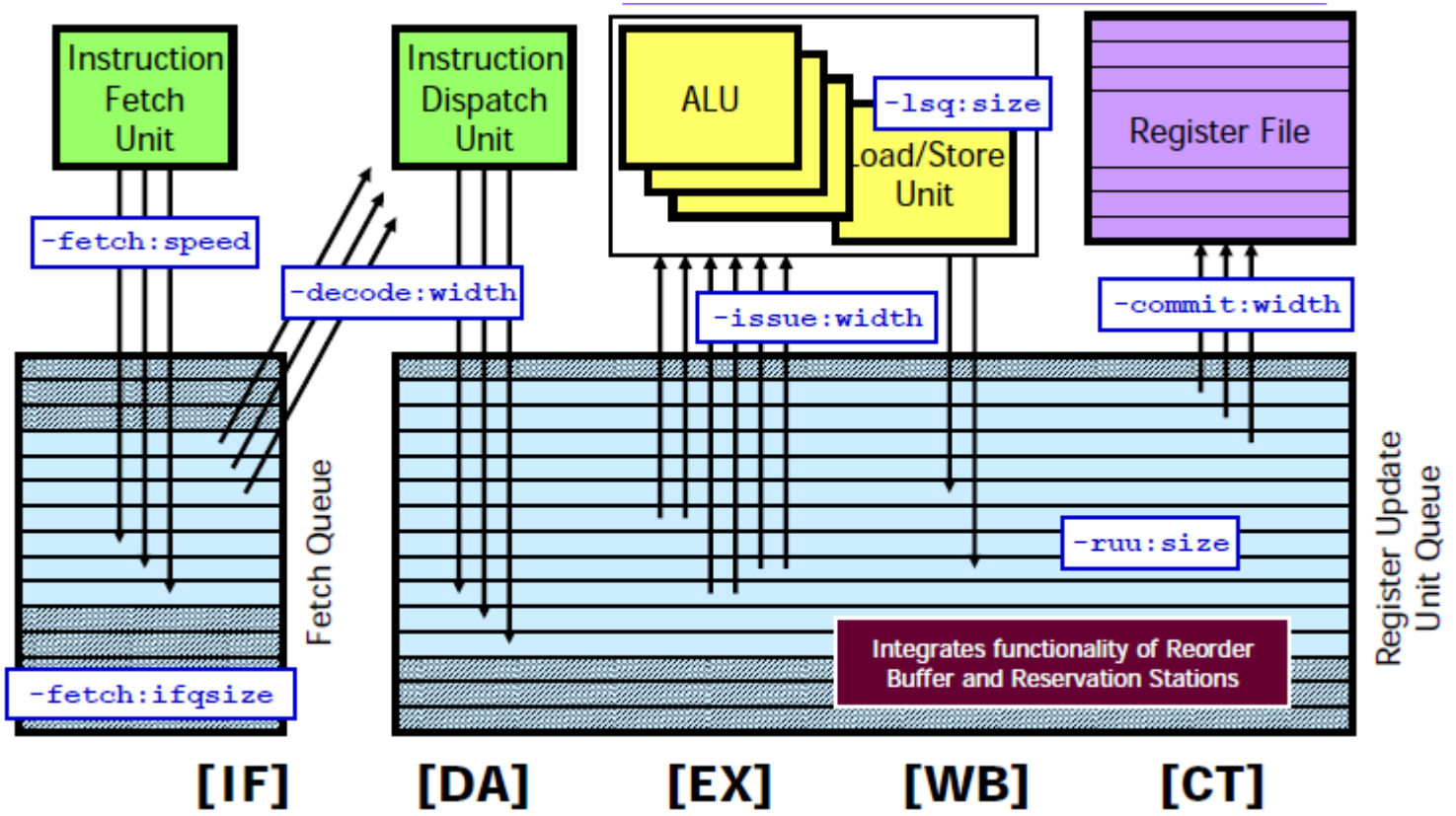
Sim-outorder

- Simulador de rendimiento detallado.
- Núcleo de ejecución out-of-order.
- Renombrado de registros, buffer de reordenamiento, ejecución especulativa.
- Jerarquía de cache de 2 niveles
- Predicción de saltos

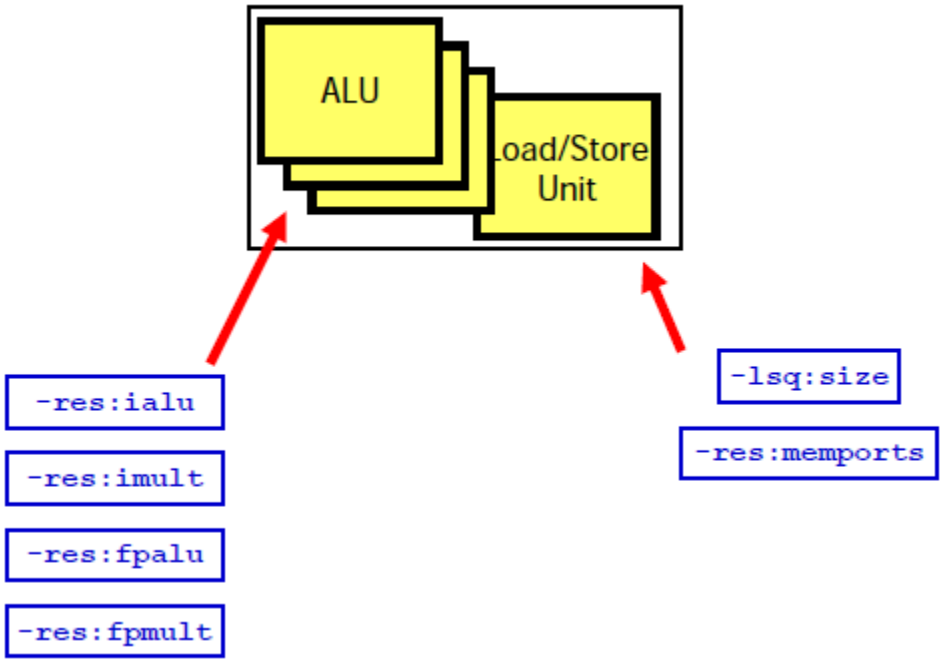
Sim-outorder



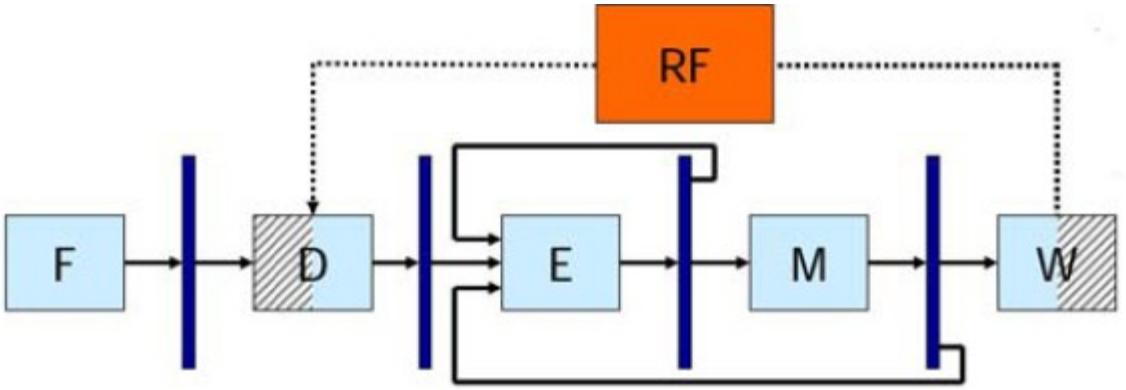
Sim-outorder



Sim-outorder



Forwarding

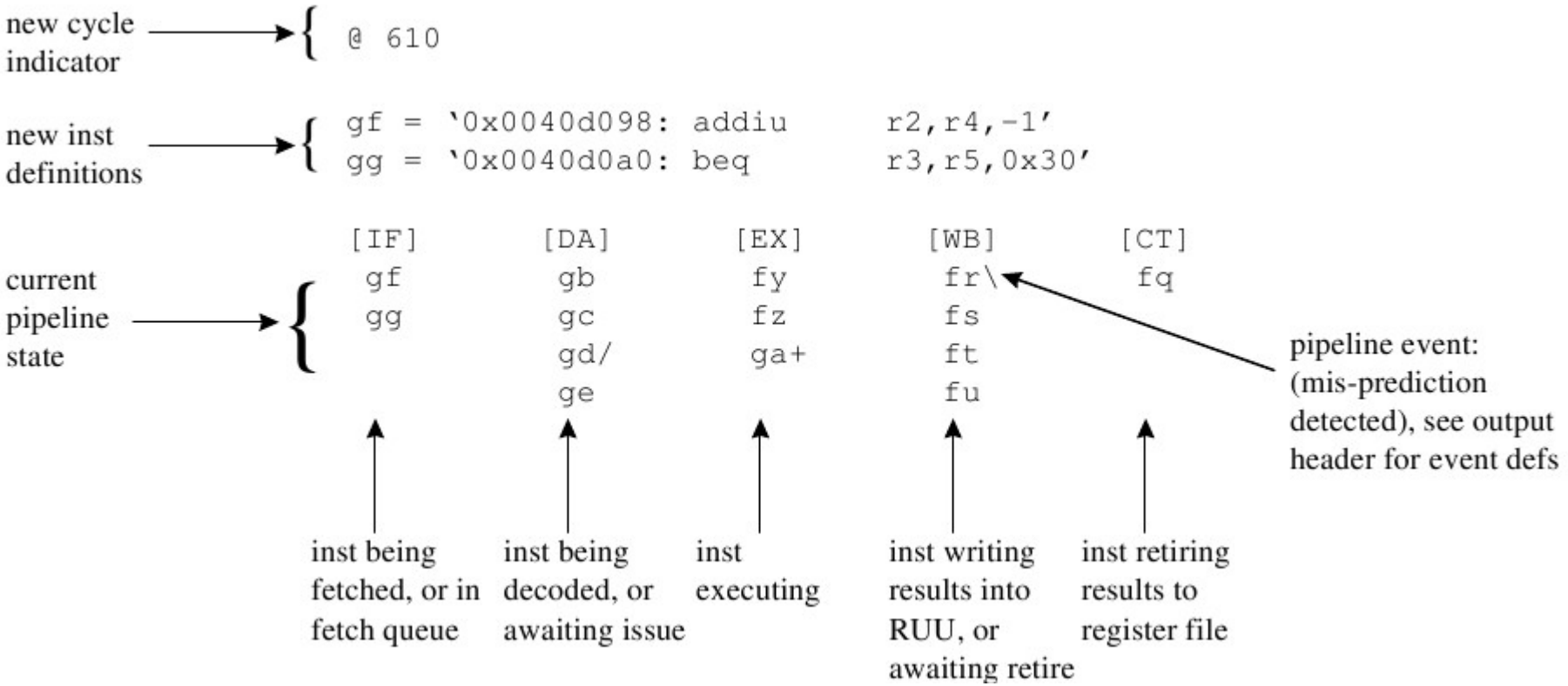


Trazas de pipeline

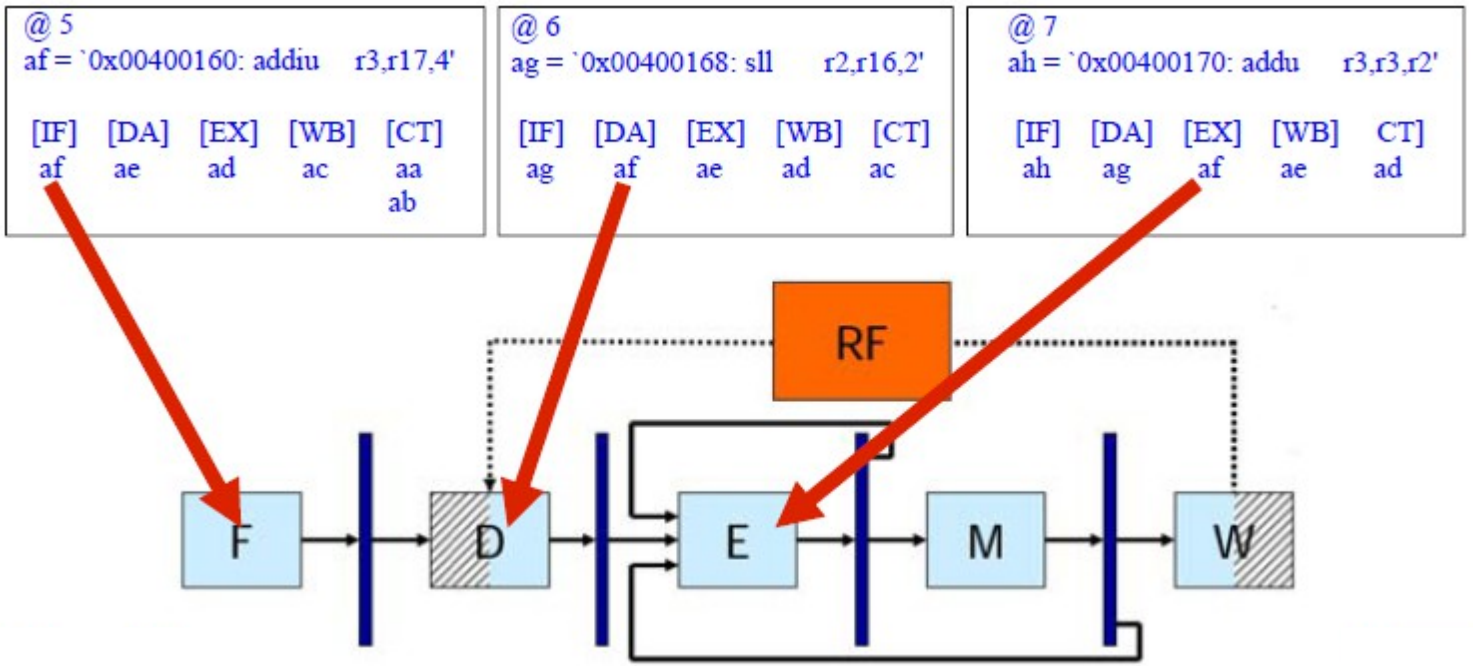
- Produce una historia detallada de todas las instrucciones ejecutadas.
- Soportado solo en sim-outorder
- Usar la opción “-ptrace”
  - -ptrace <file> <range>
- Ejemplo de uso:
  - -pcstat FOO.trc : traza de toda la ejecución
  - -pcstat BAR.trc 100:5000 de la instrucción 100 a la 5000
  - -pcstat UXXE.trc :10000 hasta la instrucción 10000
- Se puede ver con pipeview.pl. Script perl, muestra el pipeline para cada ciclo de ejecución:
  - pipeview.pl <ptrace_file>

Trazas de pipeline

- Ejemplo de uso
  - sim-outorder -ptrace FOO.trc :1000 test-math
    pipeview.pl FOO.trc

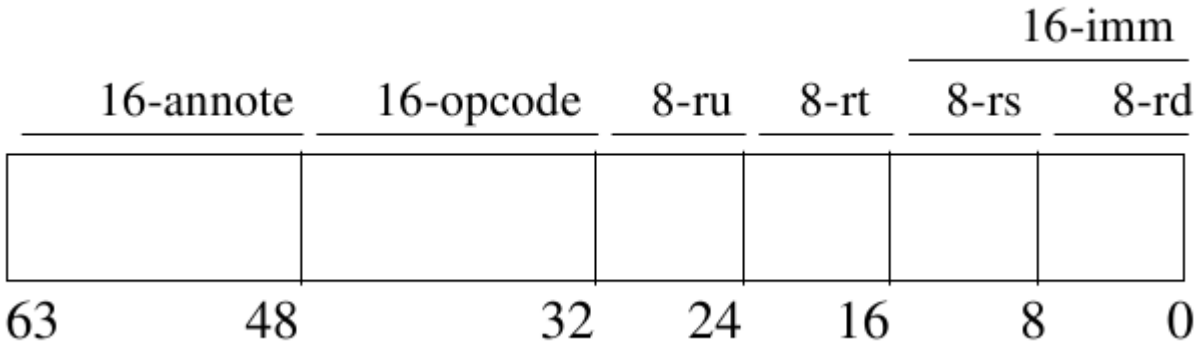


Trazas de pipeline

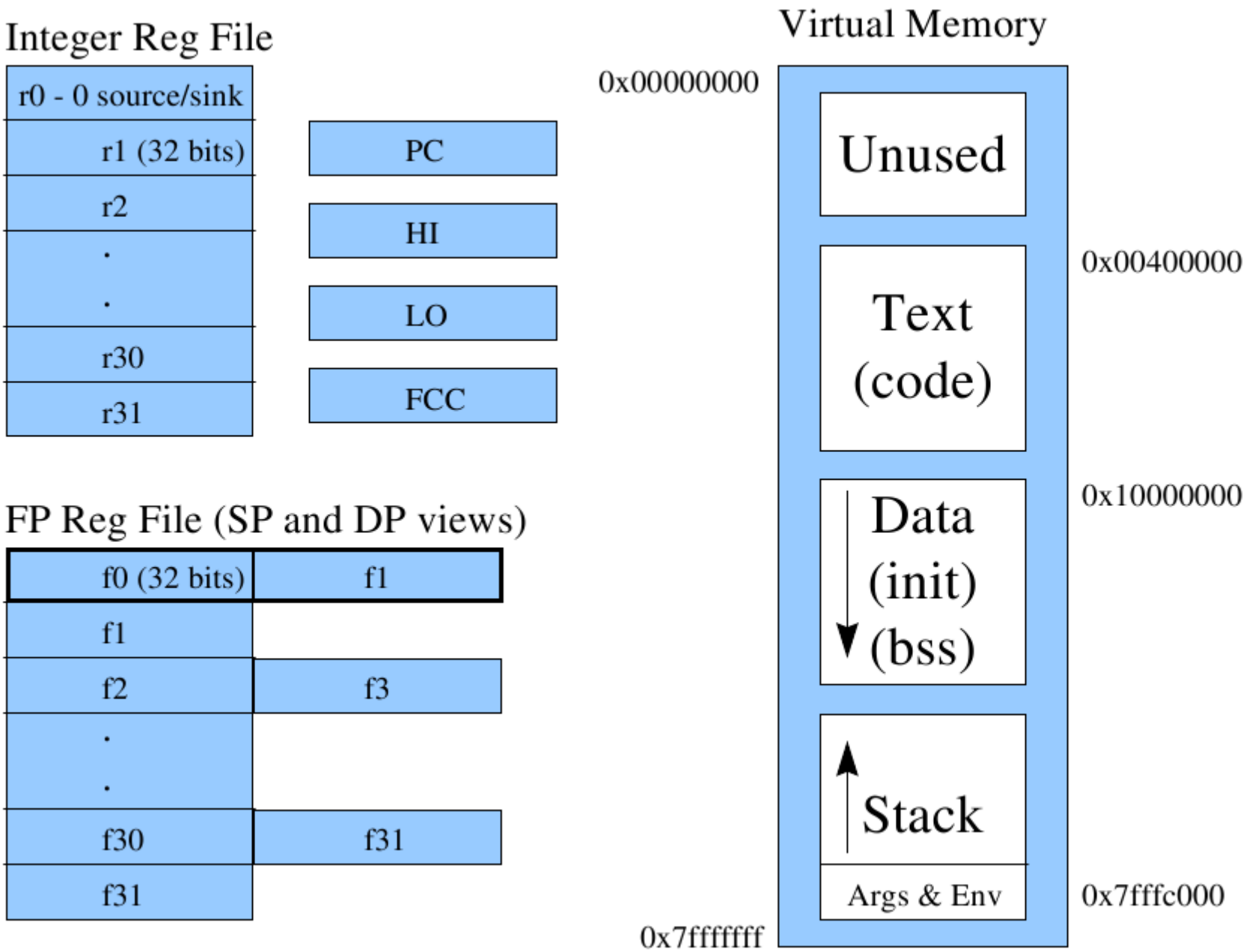


SimpleScalar ISA

- Instrucciones de 64 bits



Arquitectura SimpleScalar



Compilación y ejecución

- Ejecución
  - Para ejecutar un programa dentro del simulador (por ejemplo el programa de prueba test-math que se encuentra en tests-pisa/bin):
  - /opt/simplescalar/simplesim/sim-outorder –config config_a.cfg –ptrace config_a.trc : –redir:sim sim_configa.out ./test-math.
- Compilación de assembler PISA
  - Para compilar un programa escrito en assembler PISA debe ejecutar:
  - /opt/simplescalar/bin/sslittle-na-sstrix-gcc -nostartfiles -nostdlib -nodefaultlibs -o programa programa.s
  - Esto genera un binario con nombre “programa” a partir del archivo programa.s listo para ejecutar como se menciona en la parte anterior.