Simple Scalar v 4.0

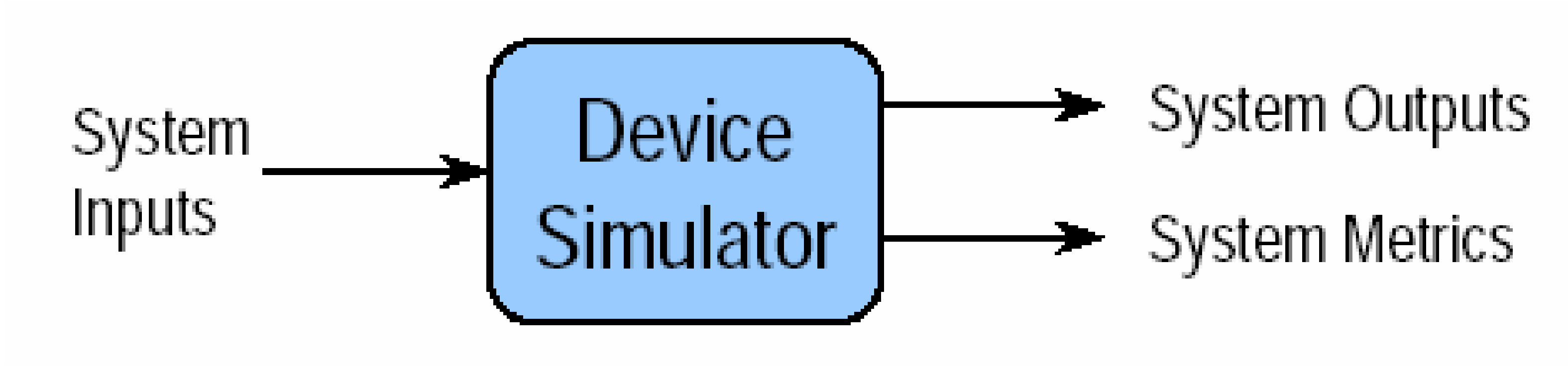
A brief introduction

Architectural simulator

- Definition: a software that reproduces the behavior of a computer system.
- A simulator is:
  - Faster in reproducing computer system behavior
  - Flexible and easier to develop
  - Easy system instrumentation
  - Easy validation considering a future hw prototype
  - Permits more design space exploration

Architectural simulator

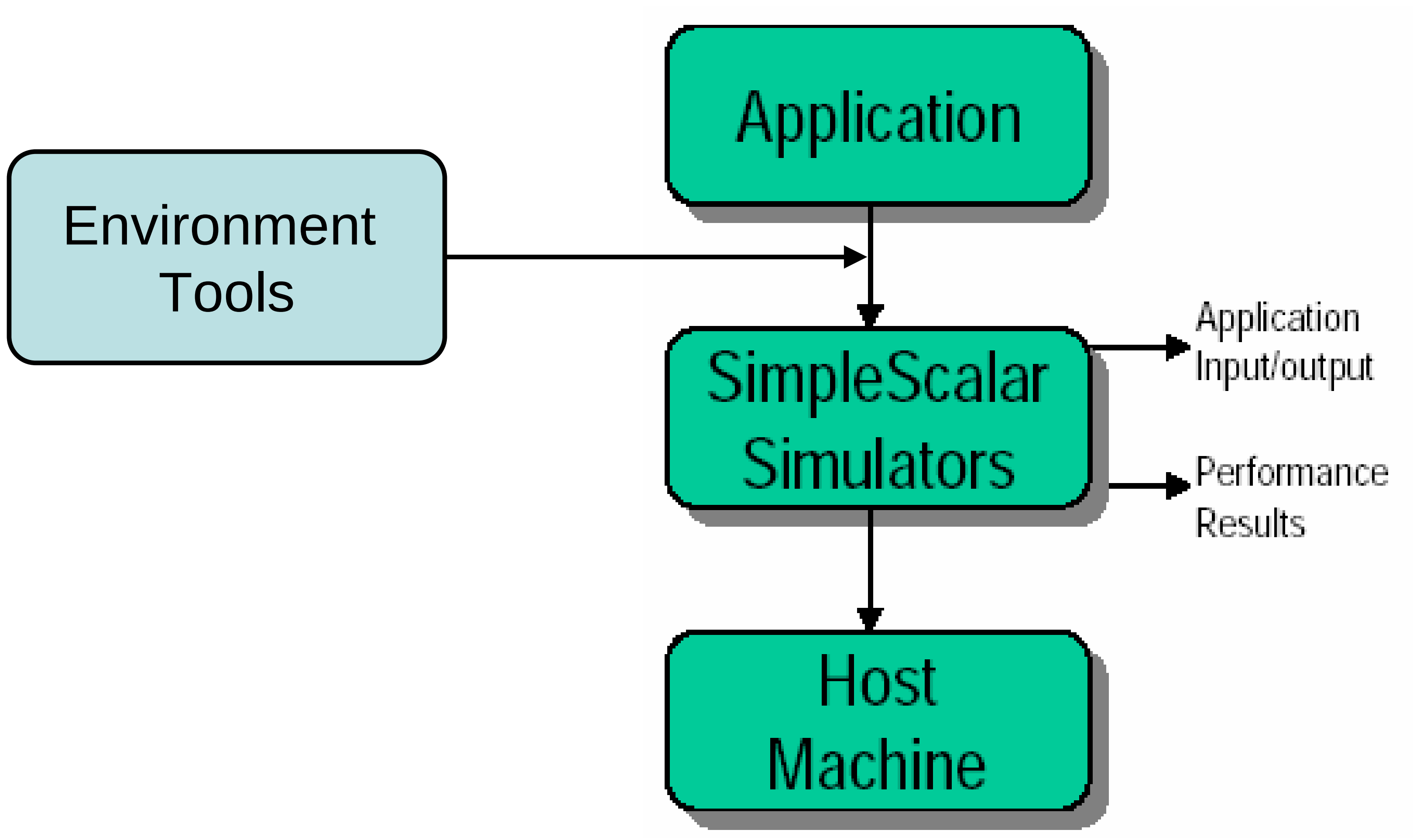
- Overall structure



Simple Scalar toolset

- The SimpleScalar toolset is not merely an hardware simulator, but a set of tools that constitutes a complete simulation environment
- Main elements:
    - Processor and devices behavioral models
    - Supports for many ISA and I/O interfaces
    - Portable to modern computer platforms
    - Rich simulation environment

Simple Scalar toolset



Simple Scalar toolset

- Miscellanea:
    - Developed at UM, UW-Madison, UT-Austin
    - Over ten year of development
    - Widely deployed in academy and industry
    - Docs and sources free available

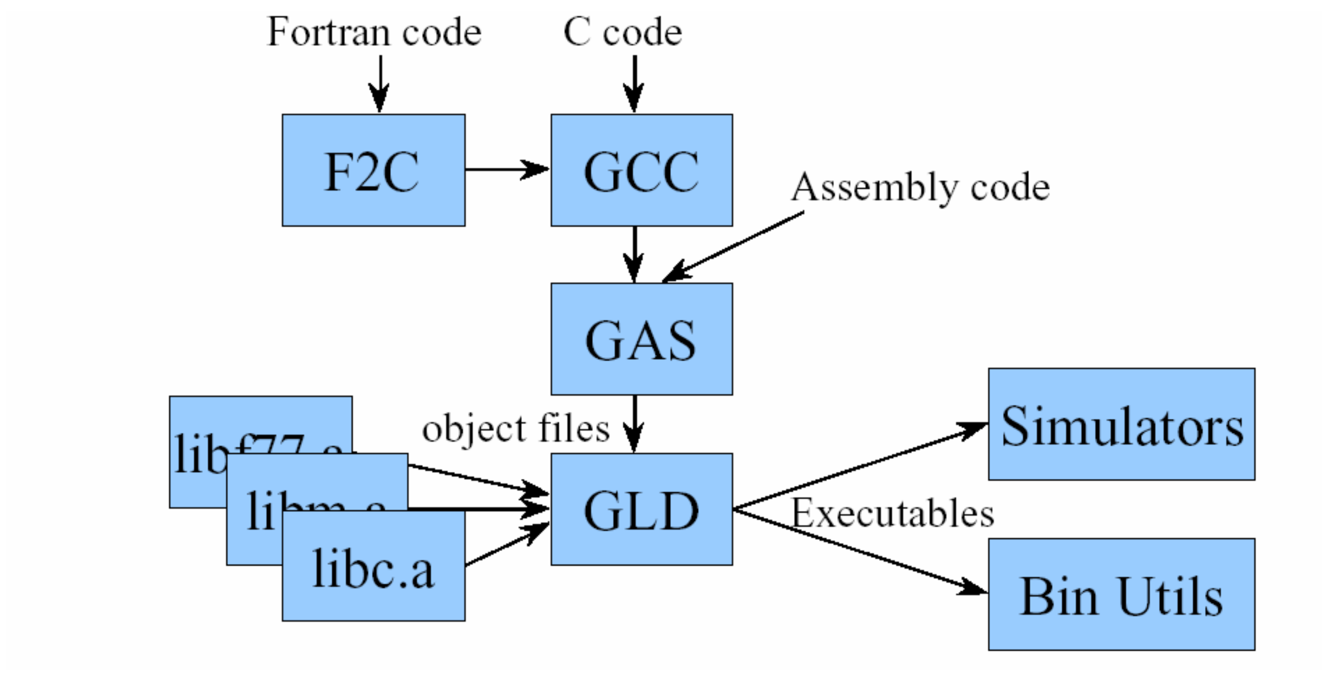
    - www.simplescalar.com
    - http://www.simplescalar.com/v4test.html

Main advantages

- Extensible
  - Source included for everything: compiler, libraries, simulators
  - Widely encoded, user-extensible instruction format
- Portable
  - At the host, virtual target runs on most Unix-like boxes
  - At the target, simulators can support multiple ISA's
- Detailed
  - Execution driven simulators
  - Supports wrong path execution, control and data speculation, etc...
  - Many sample simulators included
- Performance (on P4-1.7GHz)
  - Sim-Fast: 10+ MIPS
  - Sim-OutOrder: 350+ KIPS

Simple Scalar environment

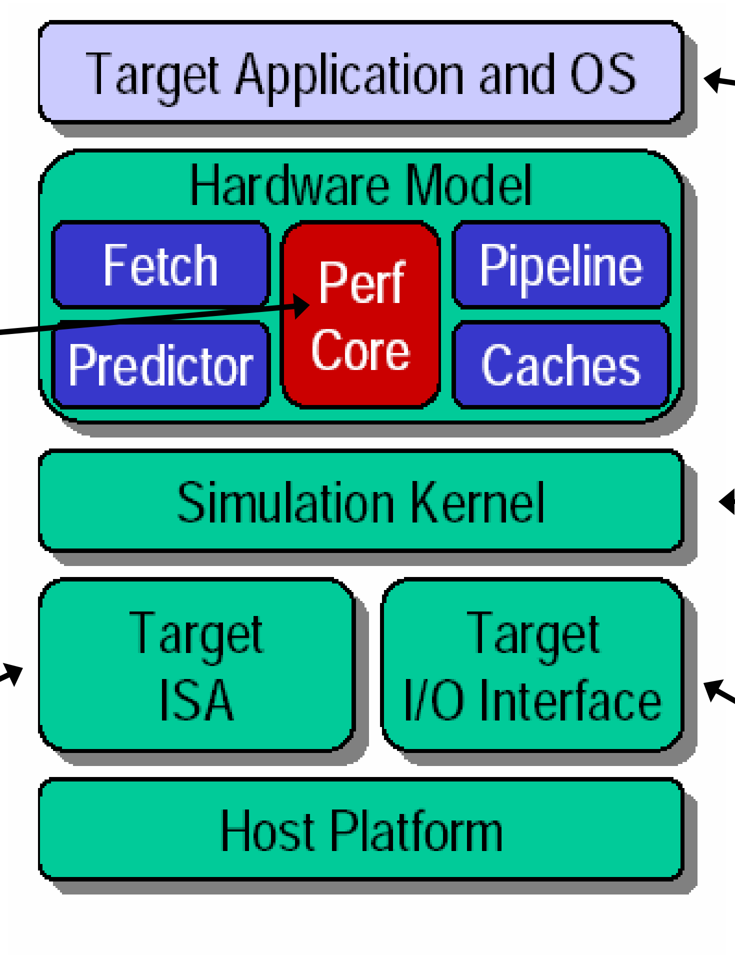
- Crosscompiler (GNU Tools)
- Libraries ported to SimpleScalar
- Third parts add-ons
- Simple Scalar Simulators



Simulator architecture

- Programming style
    - All “.c” files have an accompanying “.h” file with same base
    - “.h” files define public interfaces “exported” by module
- Mostly stable, documented with comments, studying these files
    - “.c” files implement the exported interfaces
- Simulator modules
    - sim-*.c files, each implements a complete simulator core
- Reusable S/W components facilitate “rolling your own”
    - System components
    - Simulation components
    - Add-on “really useful” components

Simulator Architecture



Applications and OS that run on simulator

Perf core implements machine

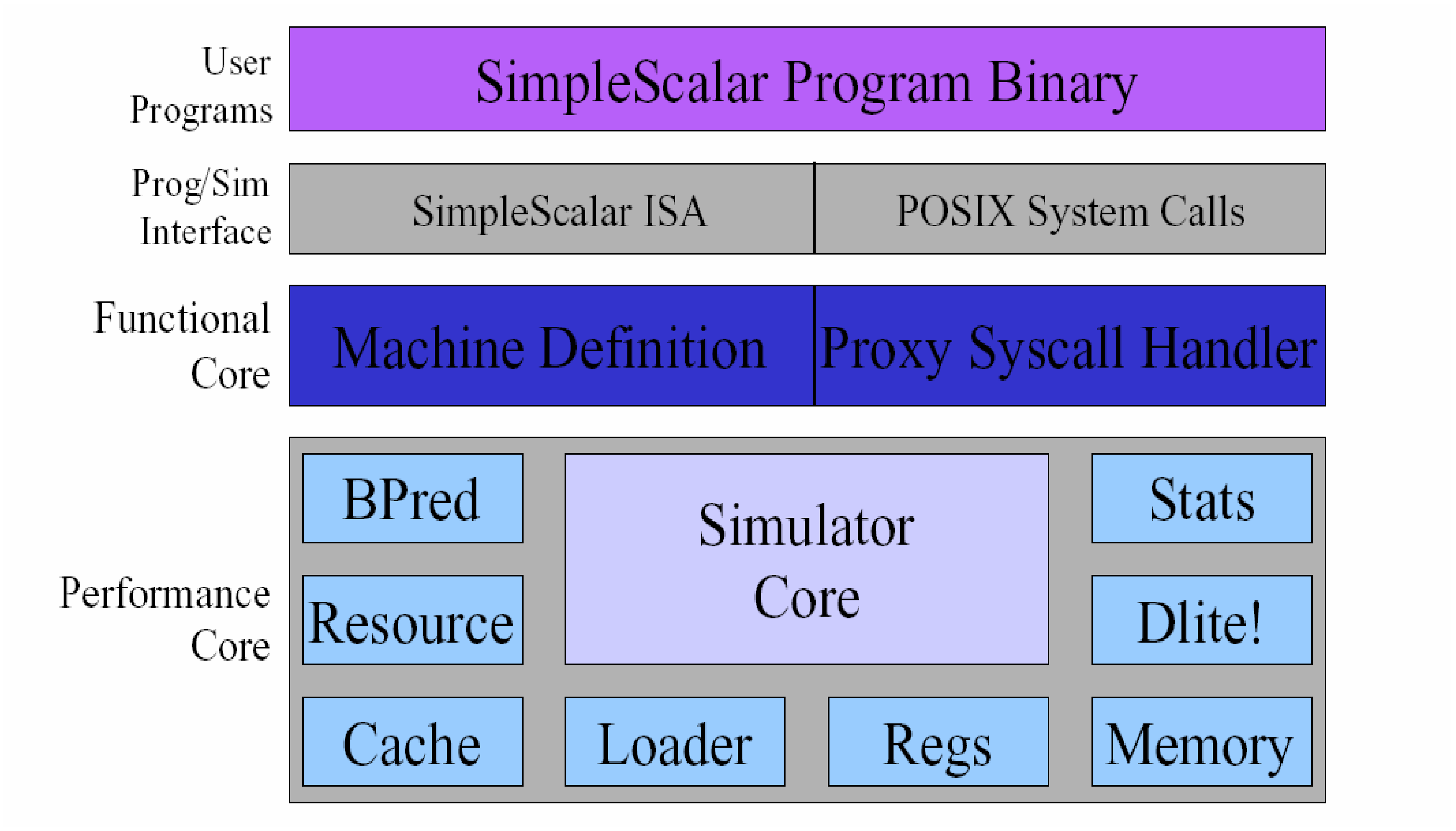
Provides event simulation services

PISA, Alpha, StrongARM, PPC, x86

Syscalls, devices, I/O traces

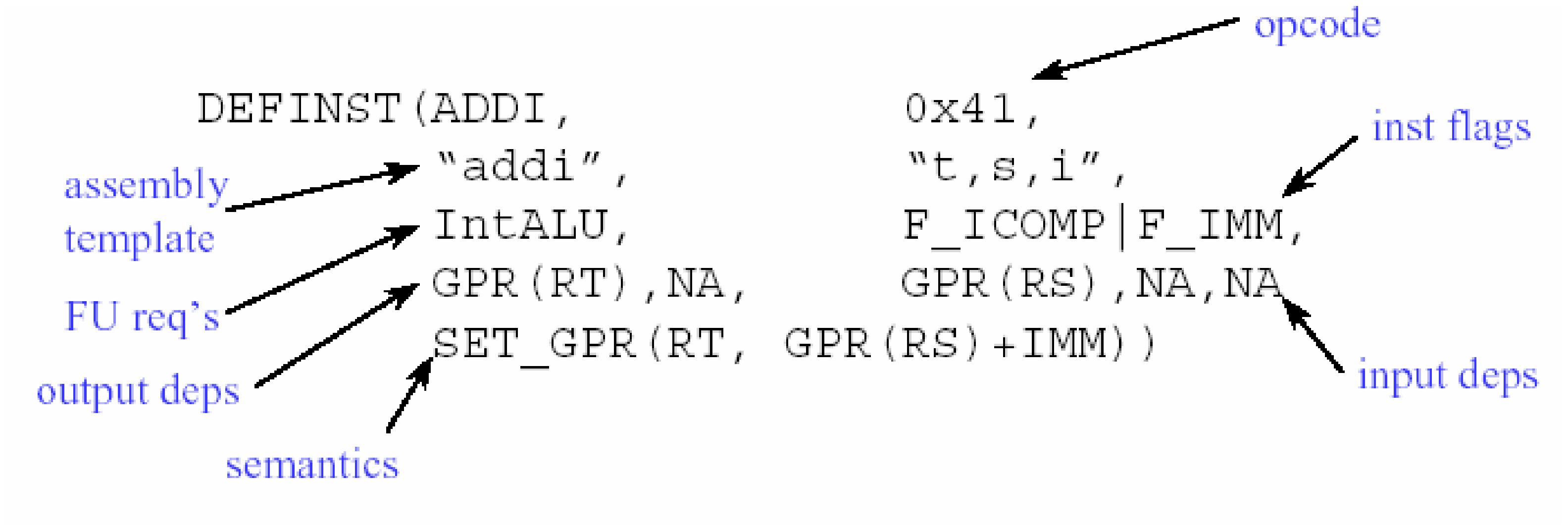
Simulator Architecture

Abstraction levels



Instruction set definition

- The instruction set architecture is described in a single file
- This file is used to build decoders, dependency analyzers, functional components, disassemblers, appendices, etc.
- Example:

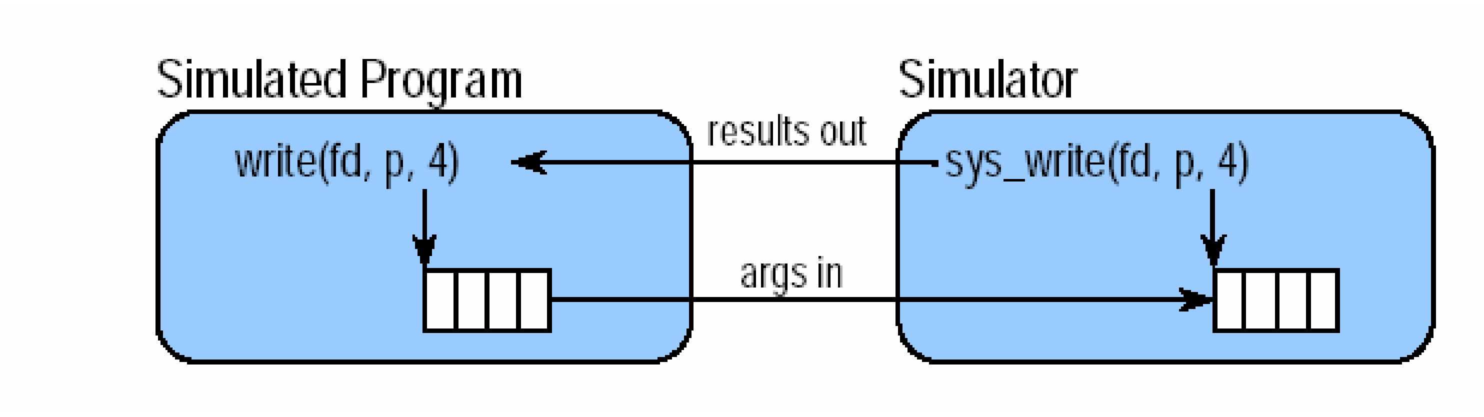


```
DEFINST(ADDI,                0x41,
        "addi",              "t,s,i",
        IntALU,              F_ICOMP|F_IMM,
        GPR(RT),NA,          GPR(RS),NA,NA
        SET_GPR(RT, GPR(RS)+IMM))
```

Simulator I/O

- To realize a useful simulator it is necessary to introduce an input/output interface:
    - I/O is supported using system calls technique
    - Ultrix syscalls are taken as inspiration example

- The system calls algo plays something like this:
    - Syscall found in target application or OS
    - Decode syscall
    - Copy syscall argument (if present) into simulator memory
    - Perform syscall on host system
    - Copy syscall result (if present) into simulator memory
    - Target application or OS can go on…

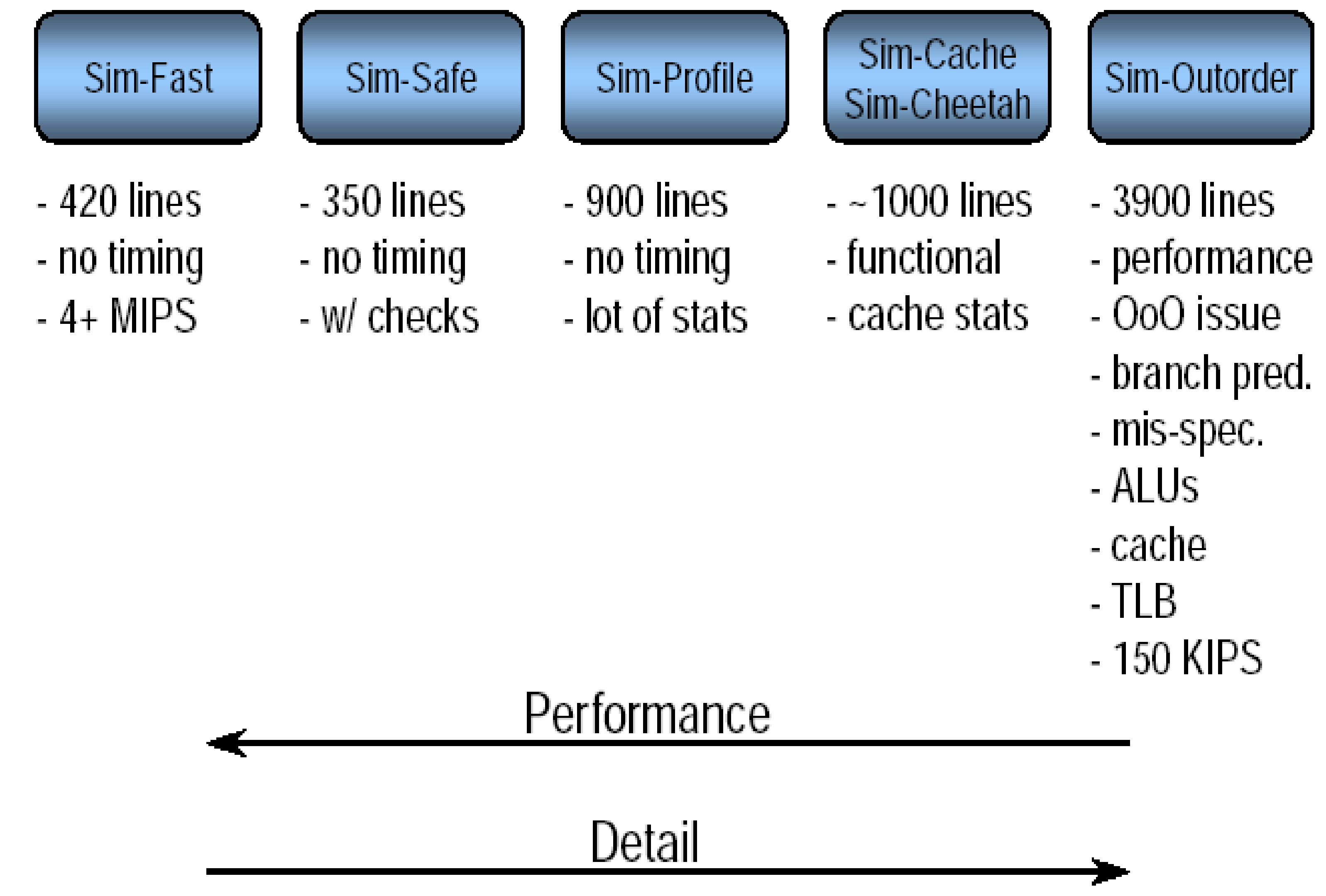


Simple Scalar simulators

- Simple Scalar environment provides seven simulators to satisfy every possible request and necessity:
  - Sim-fast
  - Sim-safe
  - Sim-profile
  - Sim-cache and sim-cheetah
  - Sim-outorder and sim-mase

- Simple Scalar simulators varies in performance, complexity, detail level and functionalities
- Every simulator is designed for a specific task but can be easily modified to satisfy specific requests

Simple Scalar simulators



Sim-Fast	Sim-Safe	Sim-Profile	Sim-Cache Sim-Cheetah	Sim-Outorder
- 420 lines	- 350 lines	- 900 lines	- ~1000 lines	- 3900 lines
- no timing	- no timing	- no timing	- functional	- performance
- 4+ MIPS	- w/ checks	- lot of stats	- cache stats	- OoO issue
				- branch pred.
				- mis-spec.
				- ALUs
				- cache
				- TLB
				- 150 KIPS

Simulators components

Standard modules

- bpred.[hc] - branch predictors
- cache.[hc] - cache module
- eventq.[hc] - event queue module
- libcheetah/ - Cheetah cache simulator library
- ptrace.[hc] - pipetrace module
- res.[hc] - resource manager module
- sim.h - simulator main code interface definitions
- textprof.pl - text segment profile view (Perl Script)
- pipeview.pl - pipetrace view (Perl script)
- dlite.[hc] - DLite!, the lightweight debugger

Simulators components

Standard modules

- eio.[hc] - external I/O tracing module
- loader.[hc] - program loader
- memory.[hc] - flat memory space module
- regs.[hc] - register module
- machine.[hc] - target and ISA-dependent routines
- machine.def - SimpleScalar ISA definition
- symbol.[hc] - symbol table module
- syscall.[hc] - proxy system call implementation

Simulators components

Standard modules

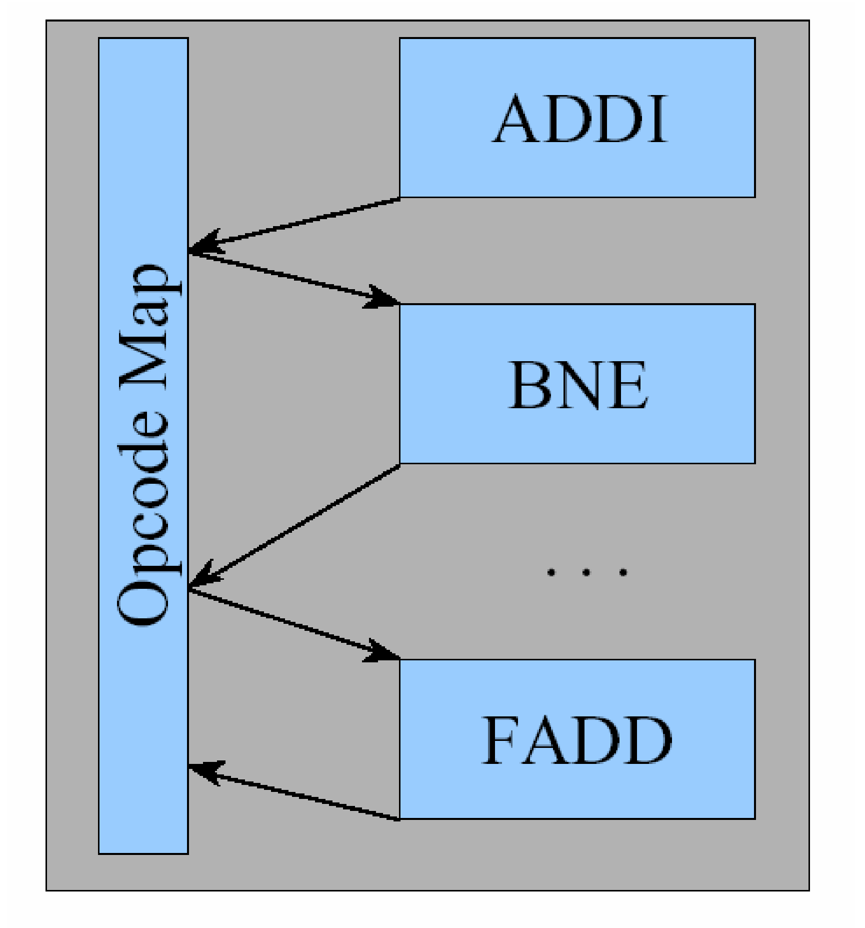
- eval.[hc] - generic expression evaluator
- libexo/ - EXO(-skeletal) persistent data structure library
- misc.[hc] - everything miscellaneous
- options.[hc] - options package
- range.[hc] - range expression package
- stats.[hc] - statistics package

Simple Scalar evolution

- The simple and earlier simulator:

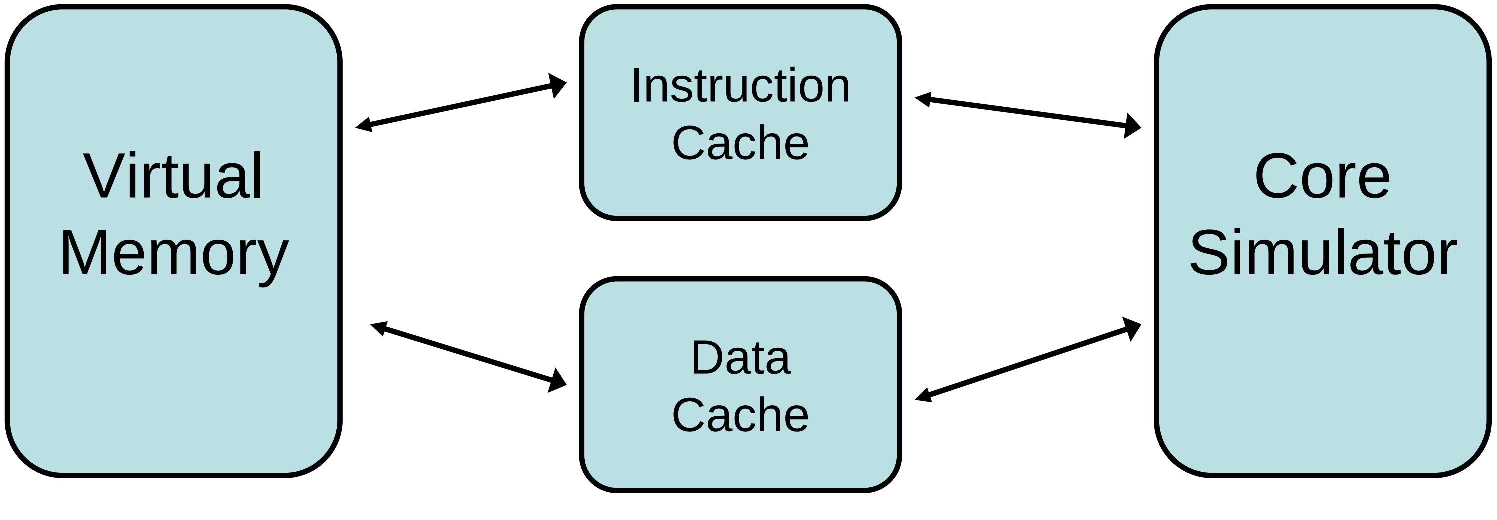
  sim-fast

- Very few options supported



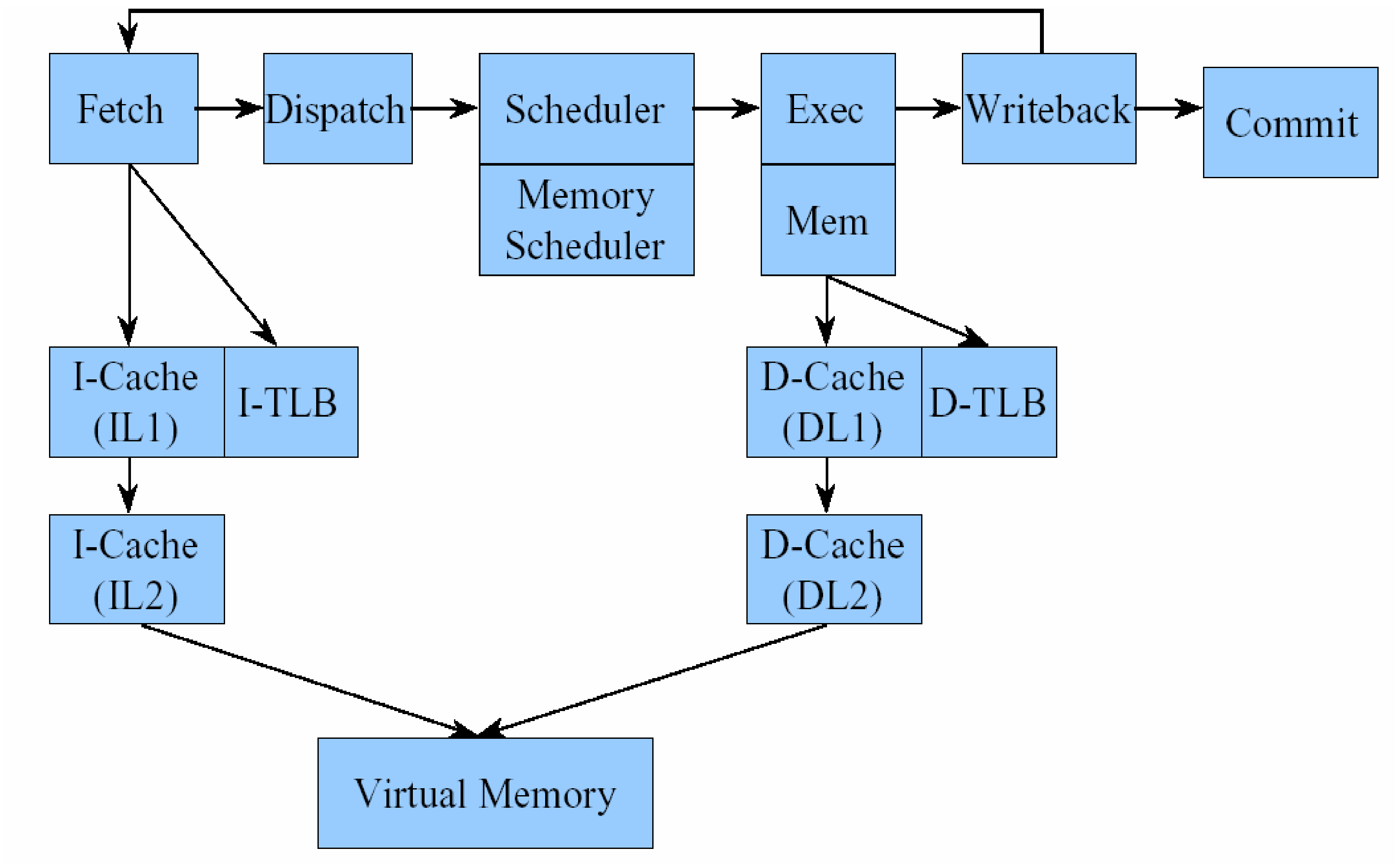
Simple Scalar evolution

- Sim-cache
  - Memory simulation
  - More feature than sim-fast
  - Profiling functions



Simple Scalar evolution

Sim-Outorder



Simple Scalar evolution

- MASE: latest evolution.
- Starting from sim-outorder have been added:
  - Micro-functional performance model
    - – Higher accuracy of the performance model
  - Checker and oracle
    - – Checker improve validation support
    - – Oracle allows for perfect studies
  - Speculative state management
    - – Simplify aggressive speculation
  - Callback interface
    - – Provides a more sophisticated memory simulation