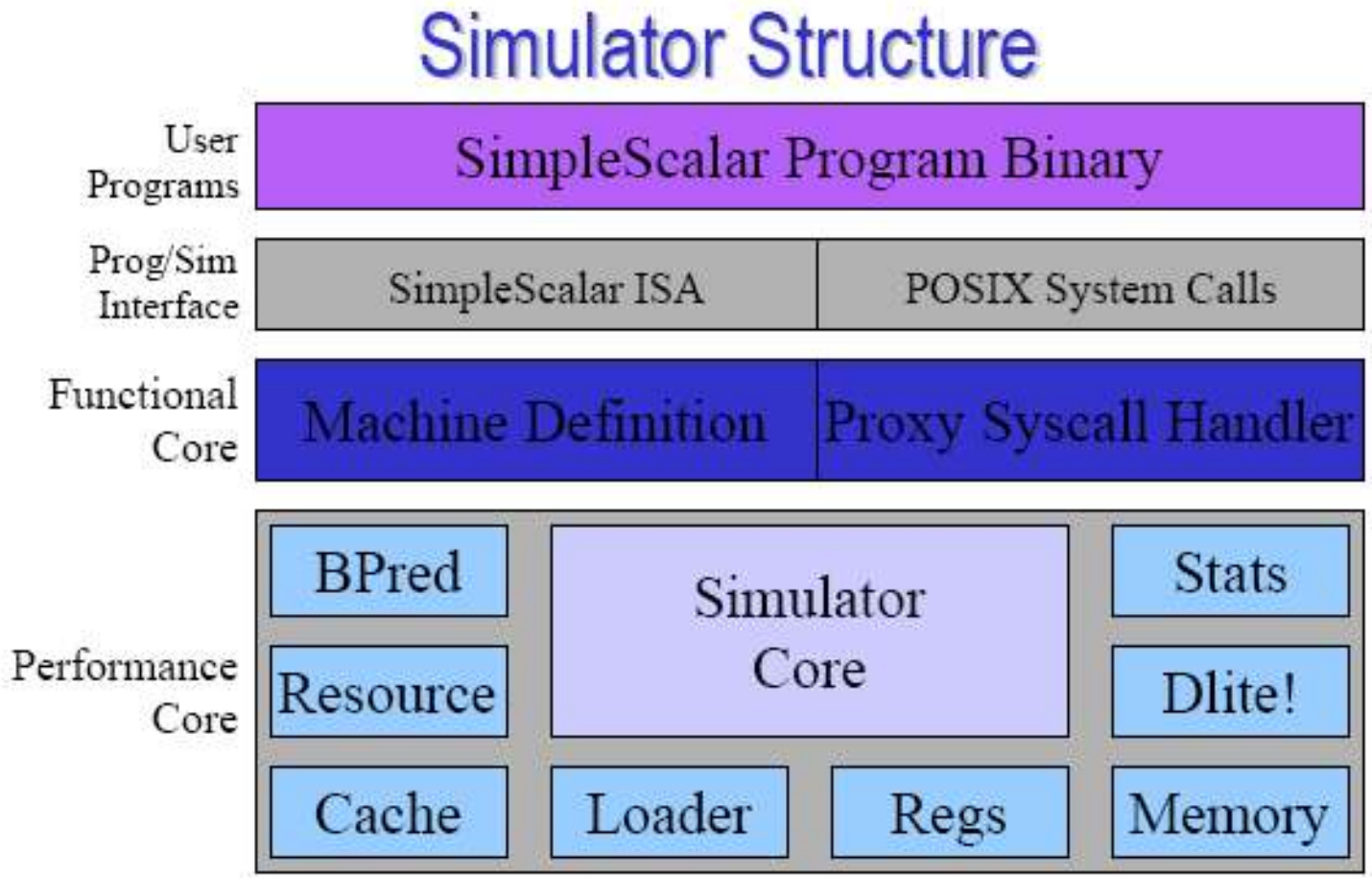
Simplescalar Tutorials

- Developed at University of Wisconsin-Madison
- Processor simulated is a derivative of the MIPS architecture
- Selectable simulators
  - Functional – fastest, least detailed
  - Cache – functional cache simulators
  - Profiling – several profiling options selected at run-time
  - Out-of-order timing – most complicated and detailed simulator, supports out-of-order issue and execution

Simplescalar

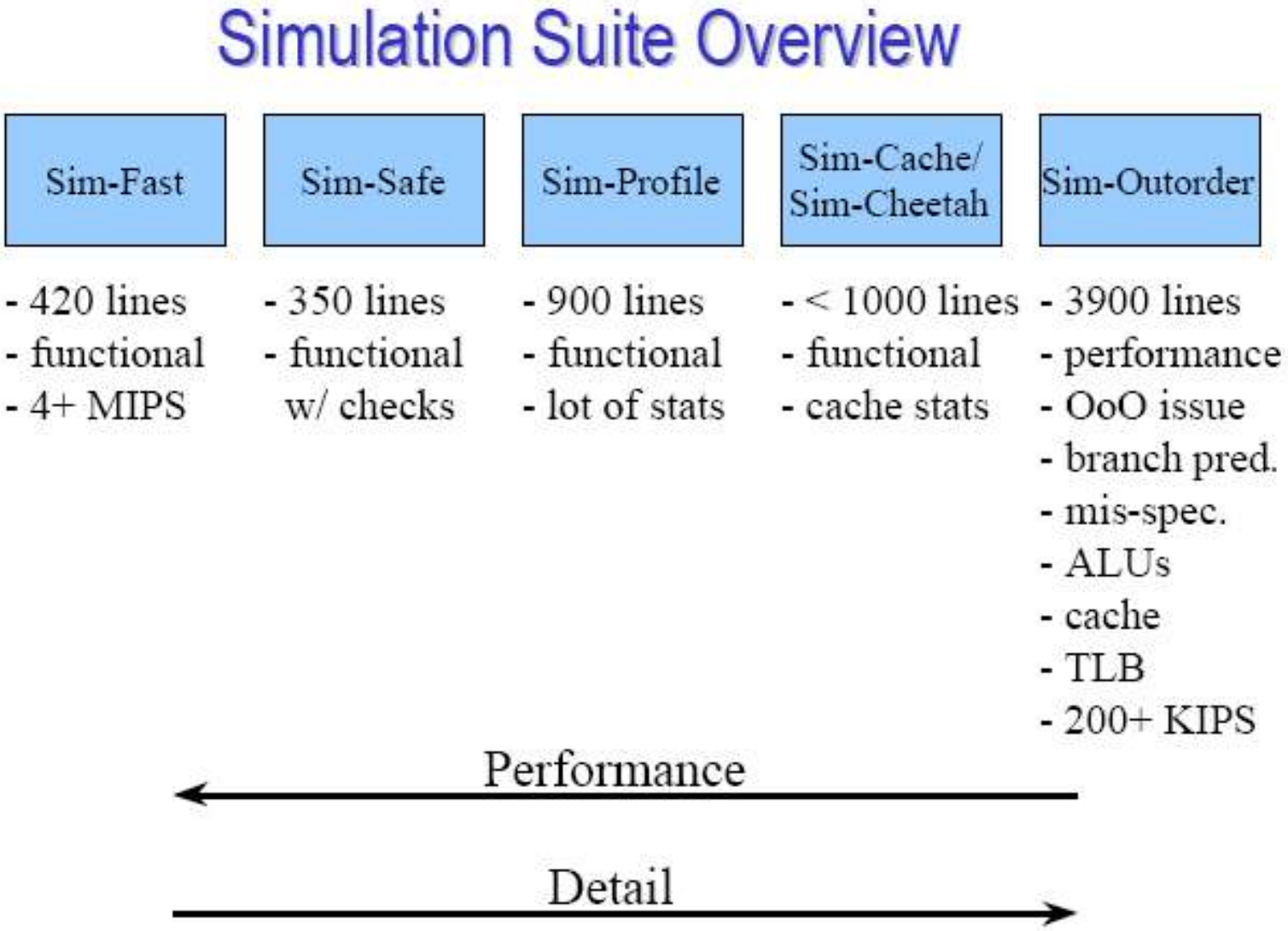
- Uniprocessor simulator
- Multiple target platforms available
- Customizable simulator
  - To add a new hardware platform, modify the source code
- Tools include a SimpleScalar specific version of GCC
  - DLite: target-machine-level debugger
  - pipeline trace viewer

Simplescalar



- modular components facilitate “rolling your own”
- performance core is optional

Simplescalar



Simplescalar

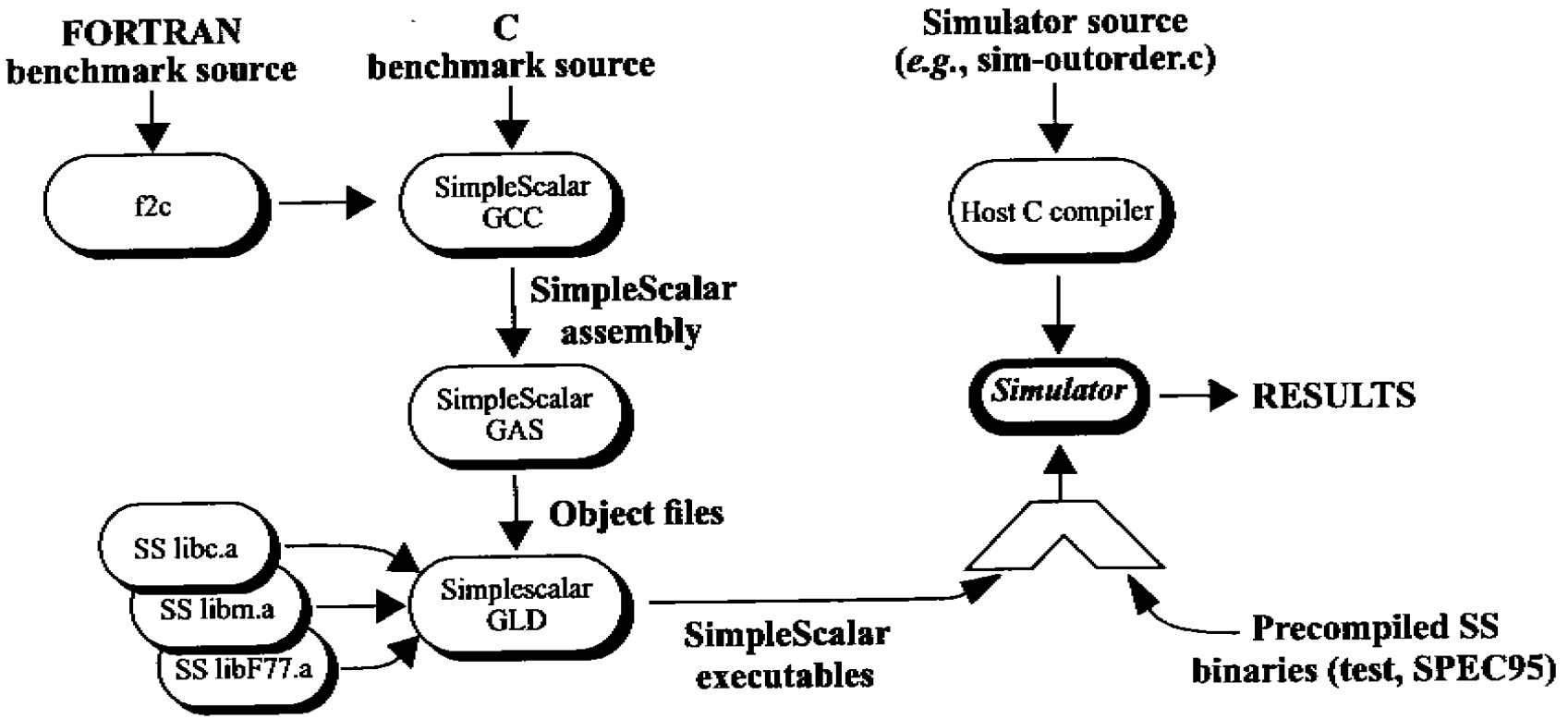


Figure 1. SimpleScalar tool set overview

How to Run

- $PATH_TO_SSLITTLE/sslittle-na-sstrix-gcc -g -o go.ss go.c
- Example for 'go' benchmark with default parameters
  - *sim-outorder -redir:sim go1.sim go.ss 9 9 go.in*
  - *Sim-outorder <options> <prog. filename> <prog. File options>*
- Always use '-redir' for file redirection
- Do not use '>' or '2>' like
  - *sim-outorder go.ss 9 9 go.in 2> output.txt //wrong*

Before Running

`-fetch:ifqsize <size>`	- instruction fetch queue size (in insts)
`-fetch:mplat <cycles>`	- extra branch mis-prediction latency (cycles)
`-bpred <type>`	- specify the branch predictor
`-decode:width <insts>`	- decoder bandwidth (insts/cycle)
`-issue:width <insts>`	- RUU issue bandwidth (insts/cycle)
`-issue:inorder`	- constrain instruction issue to program order
`-issue:wrongpath`	- permit instruction issue after mis-speculation
`-ruu:size <insts>`	- capacity of RUU (insts)
`-lsq:size <insts>`	- capacity of load/store queue (insts)
`-cache:dl1 <config>`	- level 1 data cache configuration
`-cache:dl1lat <cycles>`	- level 1 data cache hit latency

Before Running

`-cache:dl2 <config>`	- level 2 data cache configuration
`-cache:dl2lat <cycles>`	- level 2 data cache hit latency
`-cache:il1 <config>`	- level 1 instruction cache configuration
`-cache:il1lat <cycles>`	- level 1 instruction cache hit latency
`-cache:il2 <config>`	- level 2 instruction cache configuration
`-cache:il2lat <cycles>`	- level 2 instruction cache hit latency
`-cache:flush`	- flush all caches on system calls
`-cache:icompress`	- remap 64-bit inst addresses to 32-bit equiv.
`-mem:lat <1st> <next>`	- specify memory access latency (first, rest)
`-mem:width`	- specify width of memory bus (in bytes)
`-tlb:itlb <config>`	- instruction TLB configuration
`-tlb:dtlb <config>`	- data TLB configuration
`-tlb:lat <cycles>`	- latency (in cycles) to service a TLB miss

Before Running

`-res:ialu` - specify number of integer ALUs
`-res:imult` - specify number of integer multiplier/dividers
`-res:memports` - specify number of first-level cache ports
`-res:fpalu` - specify number of FP ALUs
`-res:fpmult` - specify number of FP multiplier/dividers
`-pcstat <stat>` - record statistic `<stat>` by text address
`-ptrace <file> <range>` - generate pipetrace

- all caches and TLB configurations specified with same format

  `<name>:<nsets>:<bsize>:<assoc>:<repl>`

- where

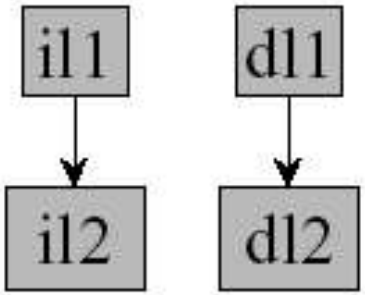
  `<name>` - cache name (make this unique)
  `<nsets>` - number of sets
  `<assoc>` - associativity (number of "ways")
  `<repl>` - set replacement policy
  `l` - for LRU
  `f` - for FIFO
  `r` - for RANDOM

- examples

  `il1:1024:32:2:l` 2-way set-assoc 64k-byte cache, LRU

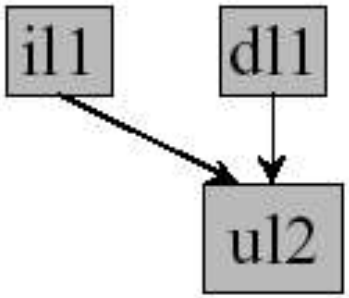
Before Running

- specify all cache parameters in no unified levels exist, e.g.,




```
-cache:il1 il1:128:64:1:l -cache:il2 il2:128:64:4:l
-cache:dl1 dl1:256:32:1:l -cache:dl2 dl2:1024:64:2:l
```

- to unify any level of the hierarchy, "point" an I-cache level into the data cache hierarchy




```
-cache:il1 il1:128:64:1:l -cache:il2 dl2
-cache:dl1 dl1:256:32:1:l -cache:dl2 ul2:1024:64:2:l
```

Examples

- Unified L1 cache and no L2.
  - *sim-outorder -cache:il1 dl1 -cache:il2 none -cache: dl1 ul1:512:64:2:l -cache:dl2 none -cache:dl1lat 6 -cache:il1lat 6 -mem:lat 18 2 -redir:sim go14.sim go.ss 9 9 go.in*
- Separate Inst and data L1 cache with no L2 along with their latencies
  - *sim-outorder -cache:il1 il1:512:128:1:l -cache:il2 none -cache:dl1 dl1:128:32:4:l -cache:dl2 none -cache:dl1lat 1 -cache:il1lat 1 -mem:lat 18 2 -redir: sim compress13.sim compress.ss < compress.in*
- Varying RUU, LSQ and Memory ports
  - *sim-outorder -ruu:size 32 -lsq:size 16 -res:memport 4 -redir:sim LSQMEMRUUgo1.sim go.ss 9 9 go.in*

Examples

- Using Branch prediction
  - sim-outorder -fetch:ifqsize 8 -decode:width 8 -issue:width 8 -commit:width 8 -ruu:size 64 -lsq:size 16 -res:memport 4 -res:ialu 6 -res:fpalu 6 -bpred:2lev 2 2048 16 0 -bpred:btb 1024 4 -redir:sim go16.sim go.ss 9 9 go.in
- Everything
  - *sim-outorder -lsq:size 16 -ruu:size 128 -fetch:ifqsize 8 -decode:width 8 -issue:width 8 -commit:width 8 -bpred:bimod 4096 -cache:il1 dl1 -cache:il2 none -cache:dl1 ul1:128:64:4:l -cache:dl2 none -cache:dl1lat 6 -cache:il1lat 6 -mem:lat 18 2 -redir:sim mempatricia8.sim patricia small.udp*
- Viewing Pipeline
  - -ptrace test.trc : - entire pipeline trace
  - -ptrace test.trc 100:1000 – trace from inst 100 to 1000
  - Finally pipeview.pl test.trc
- Benchmarks
  - DIS, Mibench, SPEC .....