TDTS10:
Computer Architecture

Lesson

2025

LINKÖPING UNIVERSITY

Outline

- Lab organization and goals
- SimpleScalar architecture and tools
- Exercises

Organization

- Assistants
  - Group A, B: Yungang Pan
  - Group C, D: Ayla Babazade
- Web page
  - http://www.ida.liu.se/~TDTS10
  - Check the lab page!

Organization

- Sign up in Webreg, deadline: Nov. 13.
- Deadline for handing in lab reports:

Lab 1	Nov. 28
Lab 2	Dec. 12
Lab 3	Dec. 19

- Rules: Read them!

Lab Schedule

Lab schedule for groups
A (by Yungang) and C (by Ayla)

Date	Time	Location*	Type
Nov. 11	13:15–15:00	A34(A), A33(C)	Lesson
Nov. 11	15:15–17:00	SU01(A), SU00(C)	Lab
Nov. 18	13:15–15:00	SU01(A), SU00(C)	Lab
Nov. 24	10:15–12:00	SU01(A), SU00(C)	Lab
Dec. 1	10:15–12:00	SU01(A), SU00(C)	Lab
Dec. 11	08:15–10:00	The resistor(A), SU00(C)	Lab
Dec. 16	13:15–15:00	The firewall(A), The back door(C)	Lab

***SU04(A) means that the students of group A should go to room SU04.**

Lab schedule for groups
B (by Yungang) and D (by Ayla)

Date	Time	Location*	Type
Nov. 11	13:15–15:00	A34(B), A33(D)	Lesson
Nov. 13	08:15–10:00	Olympus(B), SU04(D)	Lab
Nov. 21	15:15–17:00	SU01(B), SU00(D)	Lab
Nov. 27	08:15–10:00	The resistor(B), Olympus(D)	Lab
Dec. 4	08:15–10:00	SU01(B), SU00(D)	Lab
Dec. 8	10:15–12:00	The resistor(B), Olympus(D)	Lab
Dec. 15	10:15–12:00	SU01(B), SU00(D)	Lab

***SU04(B) means that the students of group B should go to room SU04.**

Please only attend your own lab sessions (6 sessions).

Please check the lab location from these tables (available on webpage).

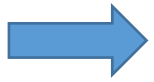
Examination

For each lab:

1. Demonstrate
   - **Must** be done during lab sessions
   - **Both** members must be present during demo
2. Report, Submitted via EMAIL
   - Subject:TDTS10-LABx-A/B/C/Dx

Labs

- Three labs:
  1. Cache Memories (2 lab sessions)
  2. Instruction Pipelining (2 lab sessions)
  3. Superscalar Processors (2 lab sessions)

Theoretical knowledge  Simulation  Analysis

Goals

- Obtain knowledge about computer organization and architecture
- Insights in various trade-offs involved in the design of a processor
- Become familiar with a set of tools necessary for evaluation of computer architectures

Environment

- Linux
- Simulations are started from a command line (i.e., terminal)
  - To open a new terminal you can press ctrl+alt+t
- Get yourself familiarized with the terminal
  - Ask Google/ChatGPT first
  - Ask your assistant
- Make sure you learn the basic commands (i.e., *cd, ls, cp*, …)
- **Make sure you are in the correct working directory**

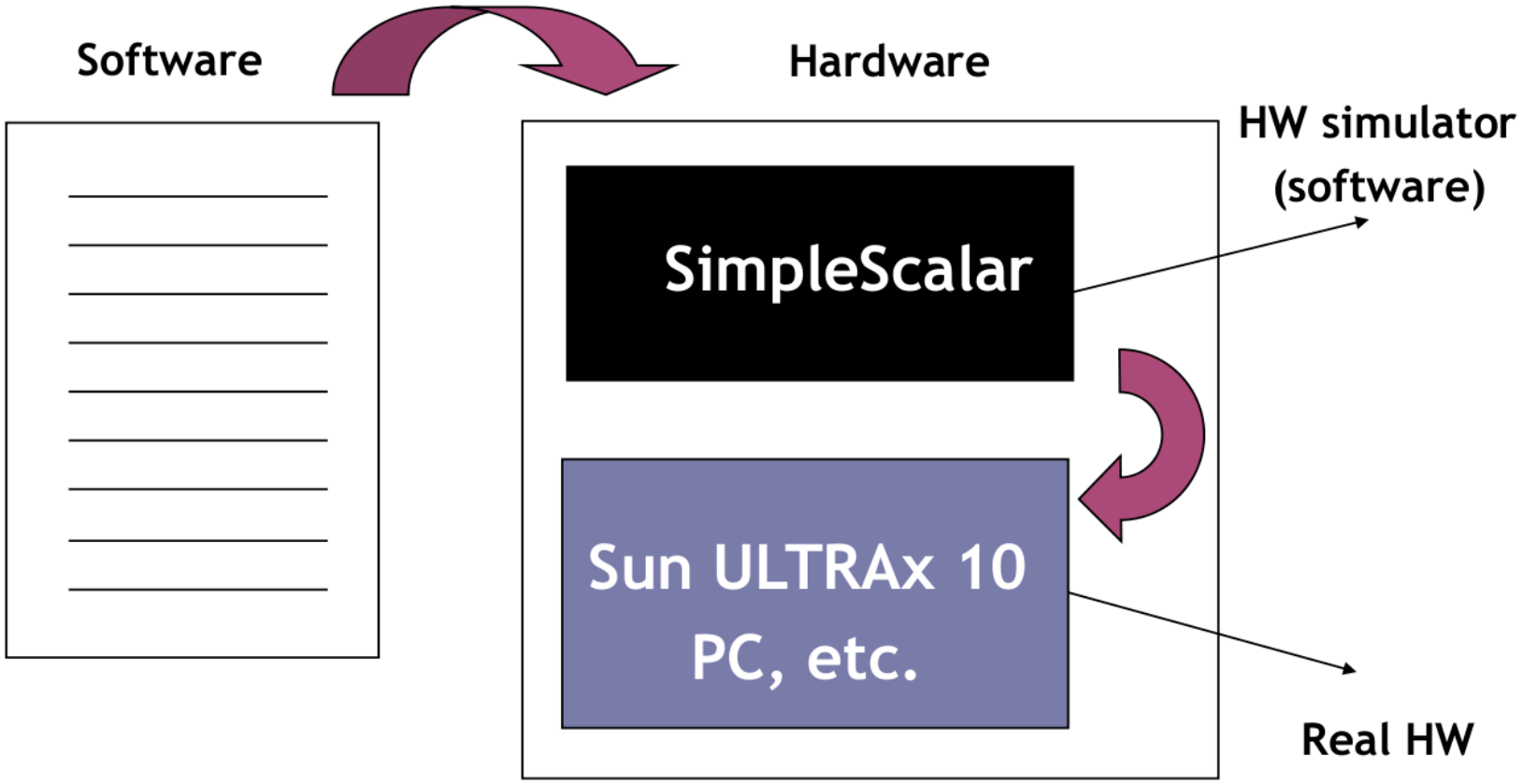
Tool Setup

- Don’t forget the instructions in lab0
- Instructions should be clear and easy to follow, but if you face difficulties
  - Don’t get frustrated :)
  - Read again carefully (without skipping over the lines)
  - Consult your assistant

Outline

- Lab organization and goals
- SimpleScalar architecture and tools
- Exercises

Architecture Simulation



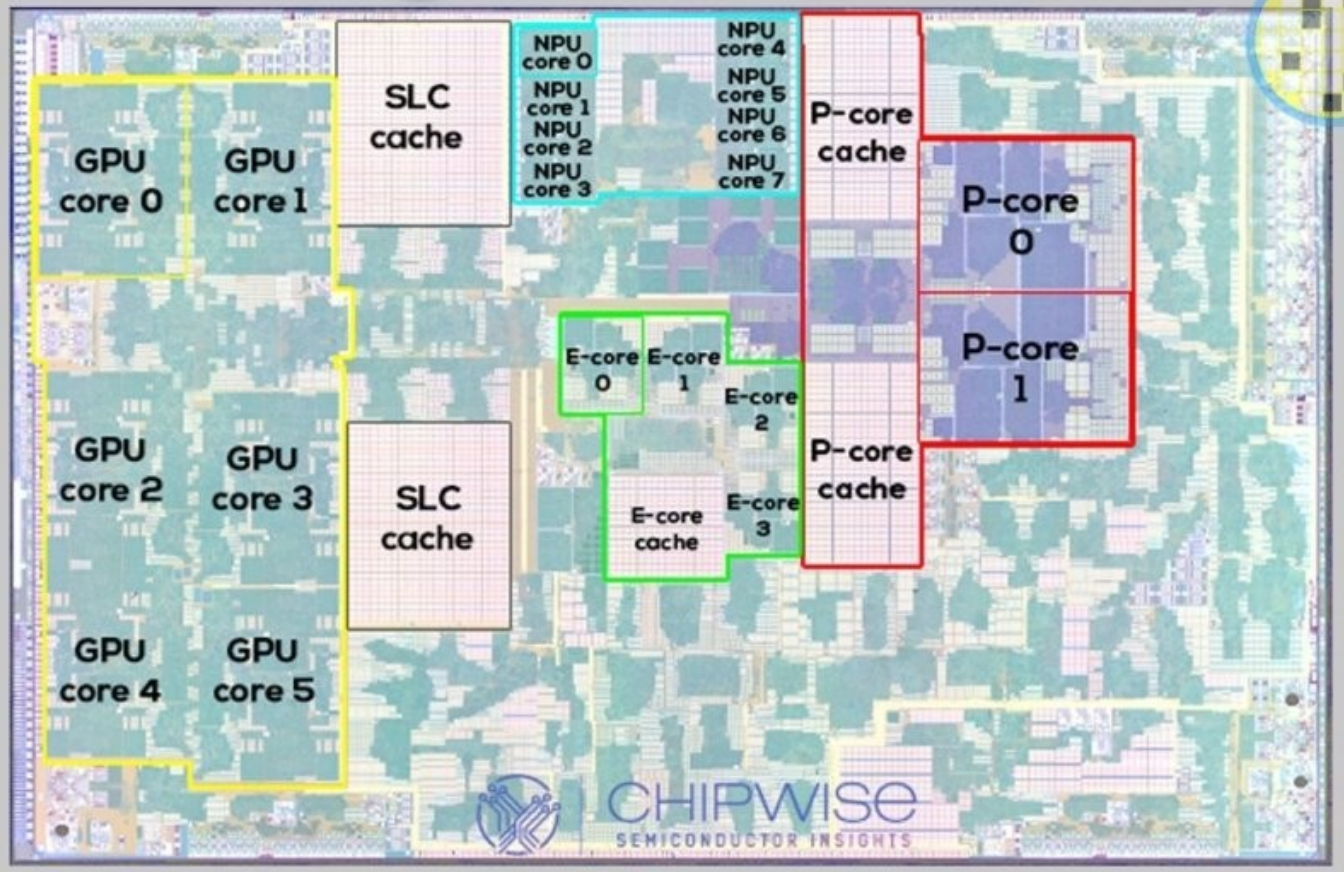
2 performance cores

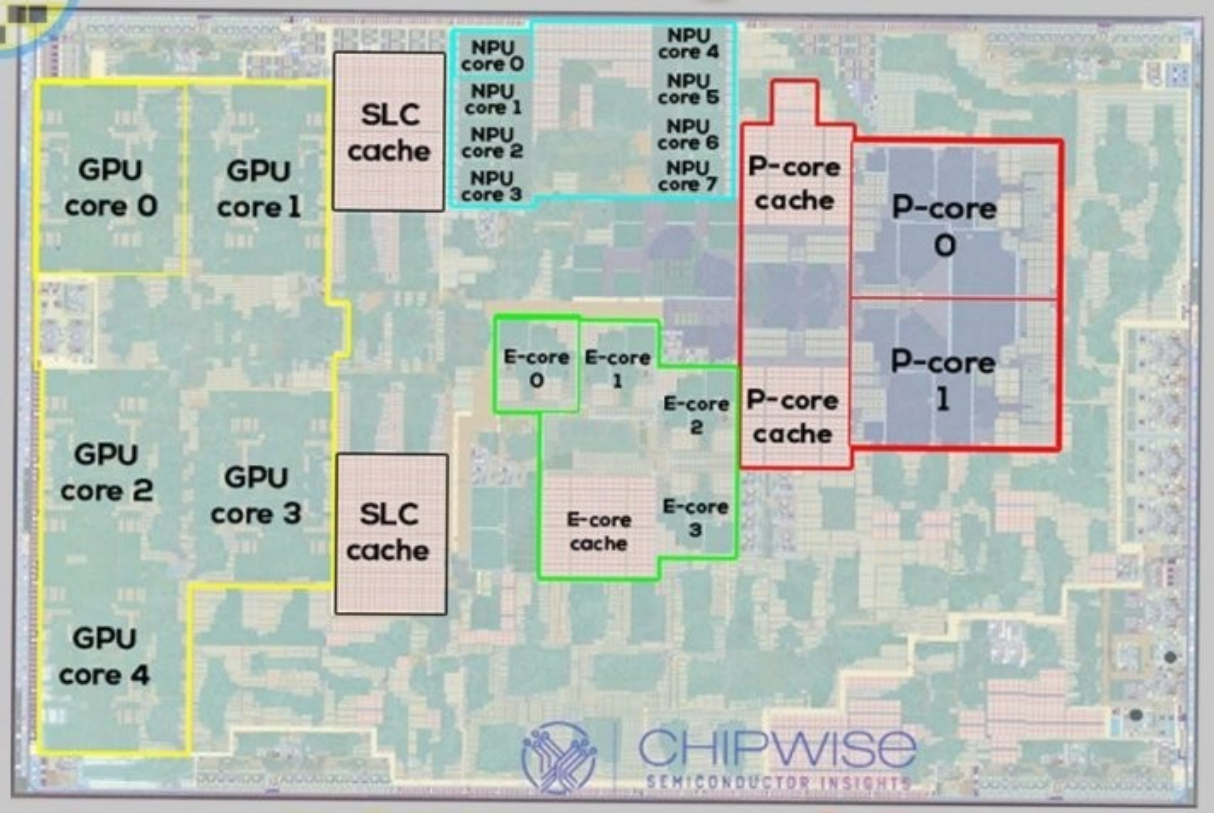
4 efficiency cores

Apple A18 Pro SoC
105mm2 @ TSMC N3E

Analysis
by HighYield

Apple A18 SoC
90mm2 @ TSMC N3E





6 GPU-cores 2 P-cores
8 NPU-cores 4 E-cores

5 GPU-cores 2 P-cores
8 NPU-cores 4 E-cores

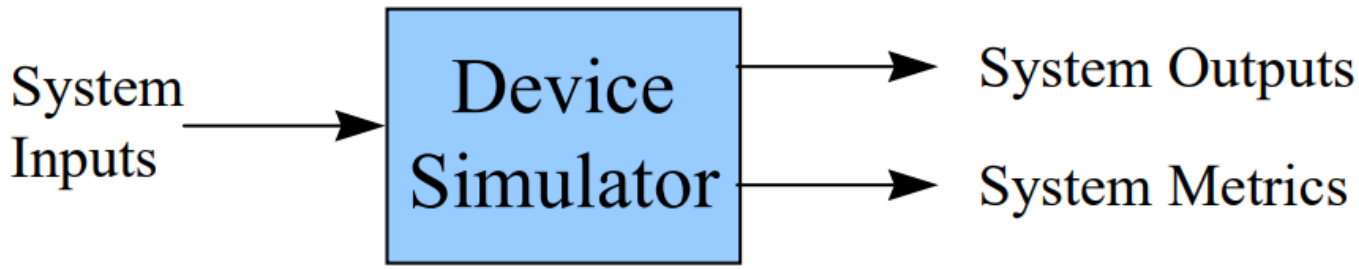
Image by ChipWise: https://chipwise.tech/our-portfolio/apple-a18-a18-pro-die-shot/

SimpleScalar: Literature

- "*The SimpleScalar Tool Set, Version 2.0*", by Doug Burger and Todd M. Austin
  - Very important preparation for the labs
  - This is your main reference for the tool!
- "User's and Hacker's guide", slides by Austin

A Computer Architecture Simulator Primer

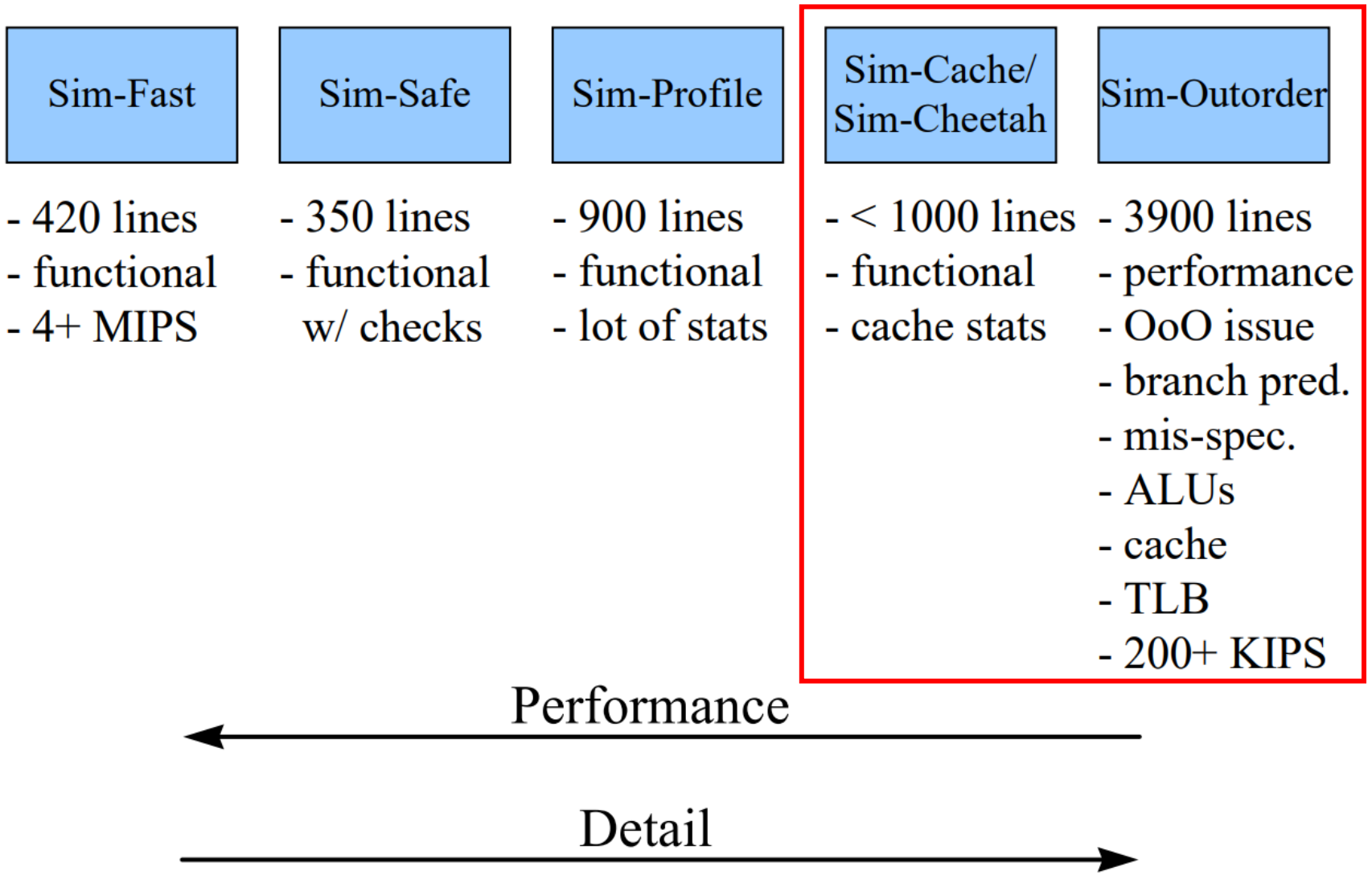
- What is an architectural simulator?
  - a tool that reproduces the behavior of a computing device



- Why use a simulator?
  - leverage faster, more flexible S/W development cycle
    - permits more design space exploration
    - facilitates validation before H/W becomes available
    - level of abstraction can be throttled to design task
    - possible to increase/improve system instrumentation

Source: https://www.cs.virginia.edu/~skadron/cs654/slides/hack_guide.pdf

Simulation Suite Overview



Source: https://www.cs.virginia.edu/~skadron/cs654/slides/hack_guide.pdf

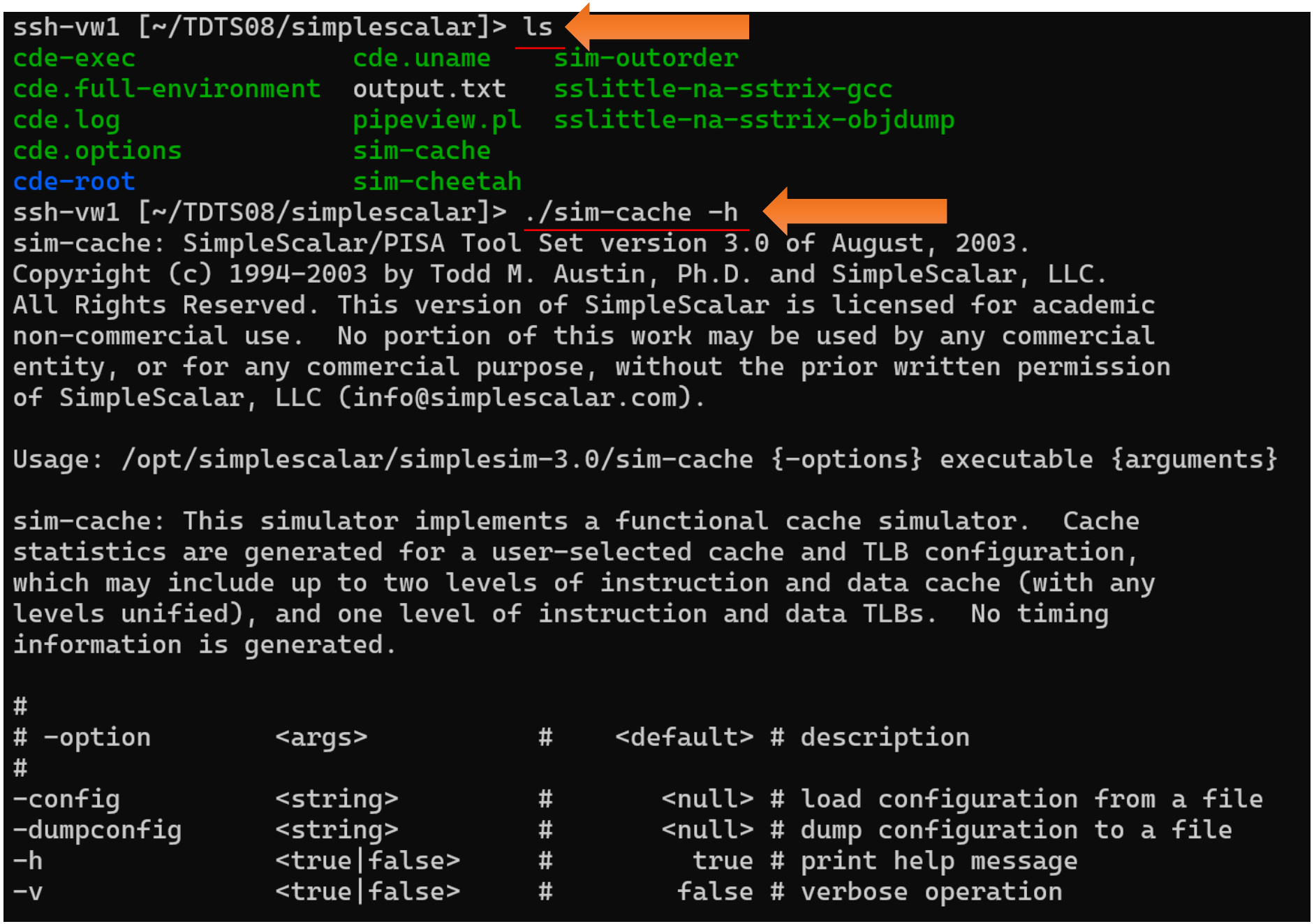
Global Simulator Options

- supported on all simulators:

`-h`	- print simulator help message
`-d`	- enable debug message
`-i`	- start up in DLite! debugger
`-q`	- terminate immediately (use with `-dumpconfig`)
`-config <file>`	- read configuration parameters from `<file>`
`-dumpconfig <file>`	- save configuration parameters into `<file>`

- configuration files:
  - to generate a configuration file:
    - specify non-default options on command line
    - and, include "`-dumpconfig <file>`" to generate configuration file
  - comments allowed in configuration files:
    - text after "#" ignored until end of line
  - reload configuration files using "`-config <file>`"
  - config files may reference other configuration files

Source: https://www.cs.virginia.edu/~skadron/cs654/slides/hack_guide.pdf


ssh-vw1 [~/TDTS08/simplescalar]> ls
cde-exec
cde.full-environment
cde.log
cde.options
cde-root
cde.uname
output.txt
pipeview.pl
sim-cache
sim-cheetah
sim-outorder
sslittle-na-sstrix-gcc
sslittle-na-sstrix-objdump
ssh-vw1 [~/TDTS08/simplescalar]> ./sim-cache -h
sim-cache: SimpleScalar/PISA Tool Set version 3.0 of August, 2003.
Copyright (c) 1994-2003 by Todd M. Austin, Ph.D. and SimpleScalar, LLC.
All Rights Reserved. This version of SimpleScalar is licensed for academic
non-commercial use. No portion of this work may be used by any commercial
entity, or for any commercial purpose, without the prior written permission
of SimpleScalar, LLC (info@simplescalar.com).
Usage: /opt/simplescalar/simplesim-3.0/sim-cache {-options} executable {arguments}
sim-cache: This simulator implements a functional cache simulator. Cache
statistics are generated for a user-selected cache and TLB configuration,
which may include up to two levels of instruction and data cache (with any
levels unified), and one level of instruction and data TLBs. No timing
information is generated.
-option <args> # <default> # description
-config <string> # <null> # load configuration from a file
-dumpconfig <string> # <null> # dump configuration to a file
-h <true|false> # true # print help message
-v <true|false> # false # verbose operation

Sim-Cache: Multi-level Cache Simulator

- generates one- and two-level cache hierarchy statistics and profiles
- extra options (also supported on sim-outorder):

`-cache:dl1 <config>` - level 1 data cache configuration
`-cache:dl2 <config>` - level 2 data cache configuration
`-cache:il1 <config>` - level 1 instruction cache configuration
`-cache:il2 <config>` - level 2 instruction cache configuration
`-tlb:dtlb <config>` - data TLB configuration
`-tlb:itlb <config>` - instruction TLB configuration
`-flush <config>` - flush caches on system calls
`-icompress` - remaps 64-bit inst addresses to 32-bit equiv.
`-pcstat <stat>` - record statistic `<stat>` by text address

Source: https://www.cs.virginia.edu/~skadron/cs654/slides/hack_guide.pdf

Specifying Cache Configurations

- all caches and TLB configurations specified with same format:

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

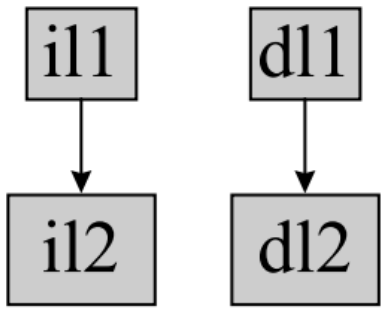
- where:
  - `<name>` - cache name (make this unique)
  - `<nsets>` - number of sets
  - `<assoc>` - associativity (number of "ways")
  - `<repl>` - set replacement policy
    - `l` - for LRU
    - `f` - for FIFO
    - `r` - for RANDOM
- examples:

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

Source: https://www.cs.virginia.edu/~skadron/cs654/slides/hack_guide.pdf

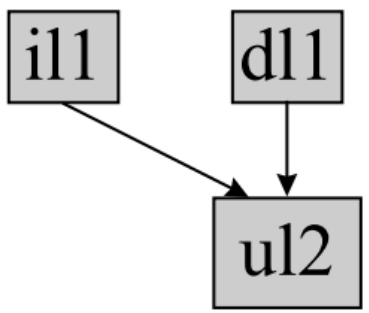
Specifying Cache Hierarchies

- 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
```

Source: https://www.cs.virginia.edu/~skadron/cs654/slides/hack_guide.pdf

Sim-Cheetah: Multi-Config Cache Simulator

- generates cache statistics and profiles for multiple cache configurations in a single program execution

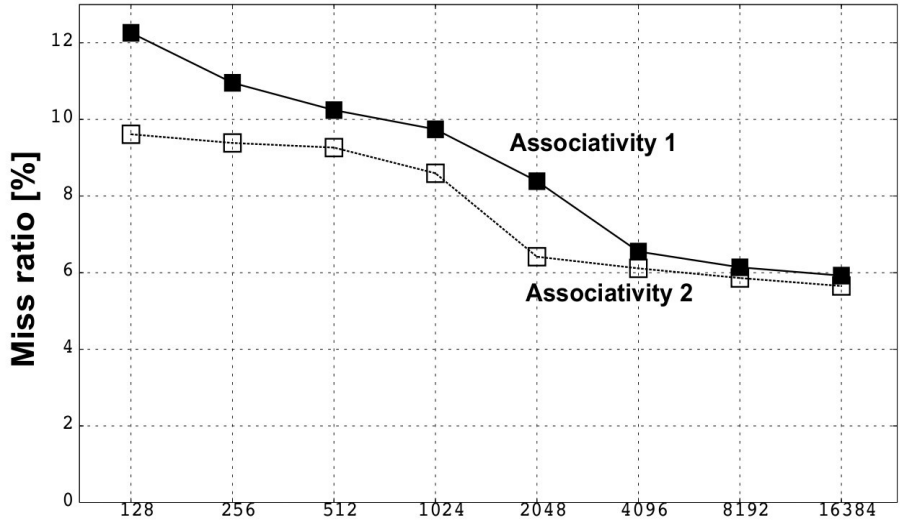
- extra options:

`-refs {inst,data,unified}`	- specify reference stream to analyze
`-C {fa,sa,dm}`	- cache config. i.e., fully or set-assoc or direct
`-R {lru, opt}`	- replacement policy
`-a <sets>`	- log base 2 number of set in minimum config
`-b <sets>`	- log base 2 number of set in maximum config
`-l <line>`	- cache line size in bytes
`-n <assoc>`	- maximum associativity to analyze (log base 2)
`-in <interval>`	- cache size interval for fully-assoc analyses
`-M <size>`	- maximum cache size of interest
`-c <size>`	- cache size for direct-mapped analyses

Source: https://www.cs.virginia.edu/~skadron/cs654/slides/hack_guide.pdf

An Example

- Lab1, assignment 3
  - Dump the default configuration of sim-cheetah
  - Modify the configuration and simulate
  - Plot the results (e.g. Matplotlib, Matlab, Excel)



Outline

- Lab organization and goals
- SimpleScalar architecture and tools
- Let’s solve some exercises on the first lab!
  - Lesson exercises