Using gem5 CPU Models

Presented by

Ayaz Akram

Outline

CPU models in gem5

AtomicSimpleCPU, TimingSimpleCPU, O3CPU, MinorCPU, KvmCPU

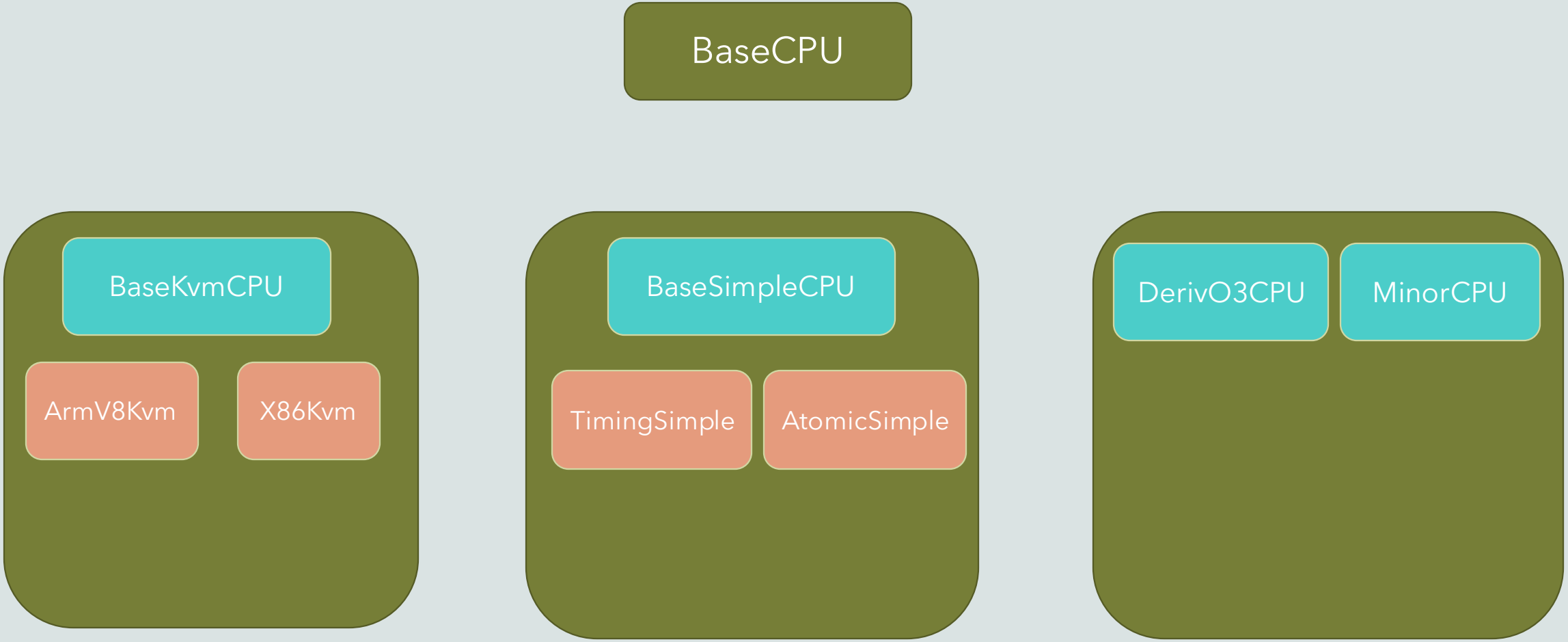
Using the CPU models

Set-up a simple system with two cache sizes and three CPU models

Look at the gem5 generated statistics

To understand differences among CPU models

Summary of gem5 CPU Models



gem5

Outline

CPU models in gem5

AtomicSimpleCPU, TimingSimpleCPU, O3CPU, MinorCPU, KvmCPU

Using the CPU models

Set-up a simple system with two cache sizes and three CPU models

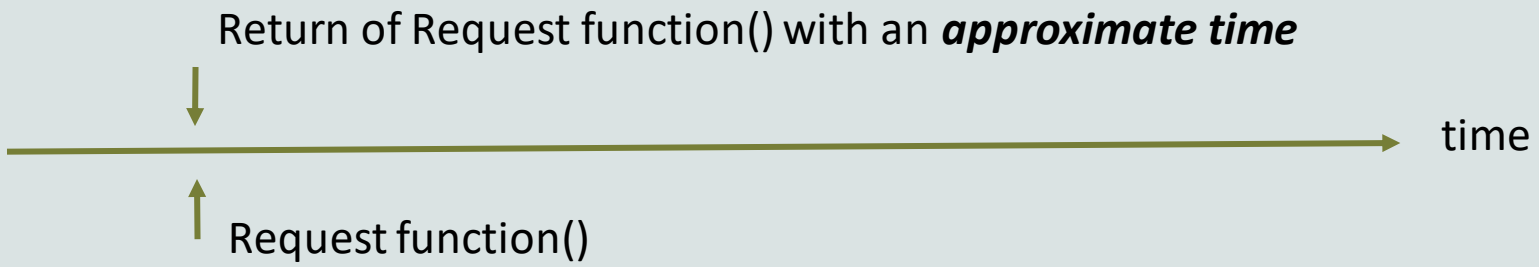
Look at the gem5 generated statistics

To understand differences among CPU models

SimpleCPU

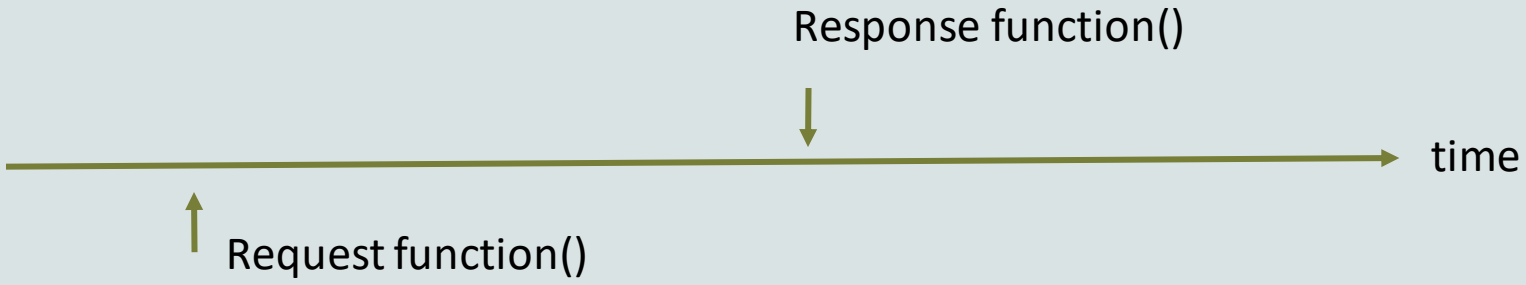
Functional, In-Order CPU Models

Atomic
Seq. of nested calls
Use: Warming up, FF



Functional
Backdoor access to mem.
(loading binaries)
No effect on coherency states

Timing
Split transactions
Models queuing delay and
resource contention



gem5

O3CPU (Out of Order CPU Model)

Timing memory accesses

execute-in-execute semantics

Time buffers between stages

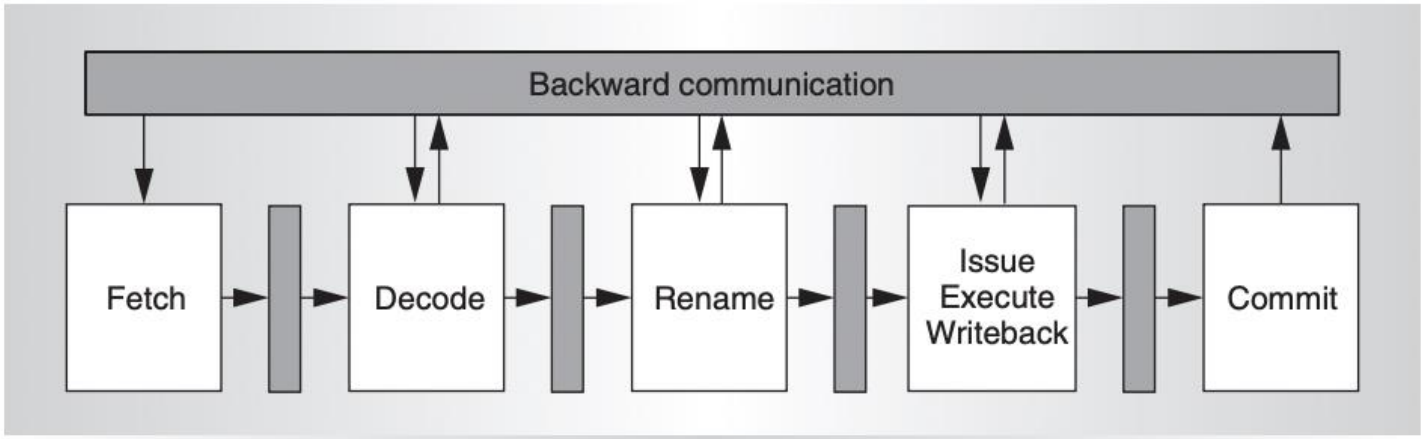


Figure 2. O3CPU pipeline. Shaded boxes represent time buffers.

AtomicSimpleCPU

Uses ***Atomic*** memory accesses

- no resource contentions or queuing delay
- Mostly used for fast-forwarding and warming of caches

TimingSimpleCPU

Uses ***Timing*** memory accesses

- Execute non-memory operations in one cycle
- Models the timing of memory accesses in detail

O3CPU Model Parameters (very configurable)

src/cpu/o3/BaseO3CPU.py

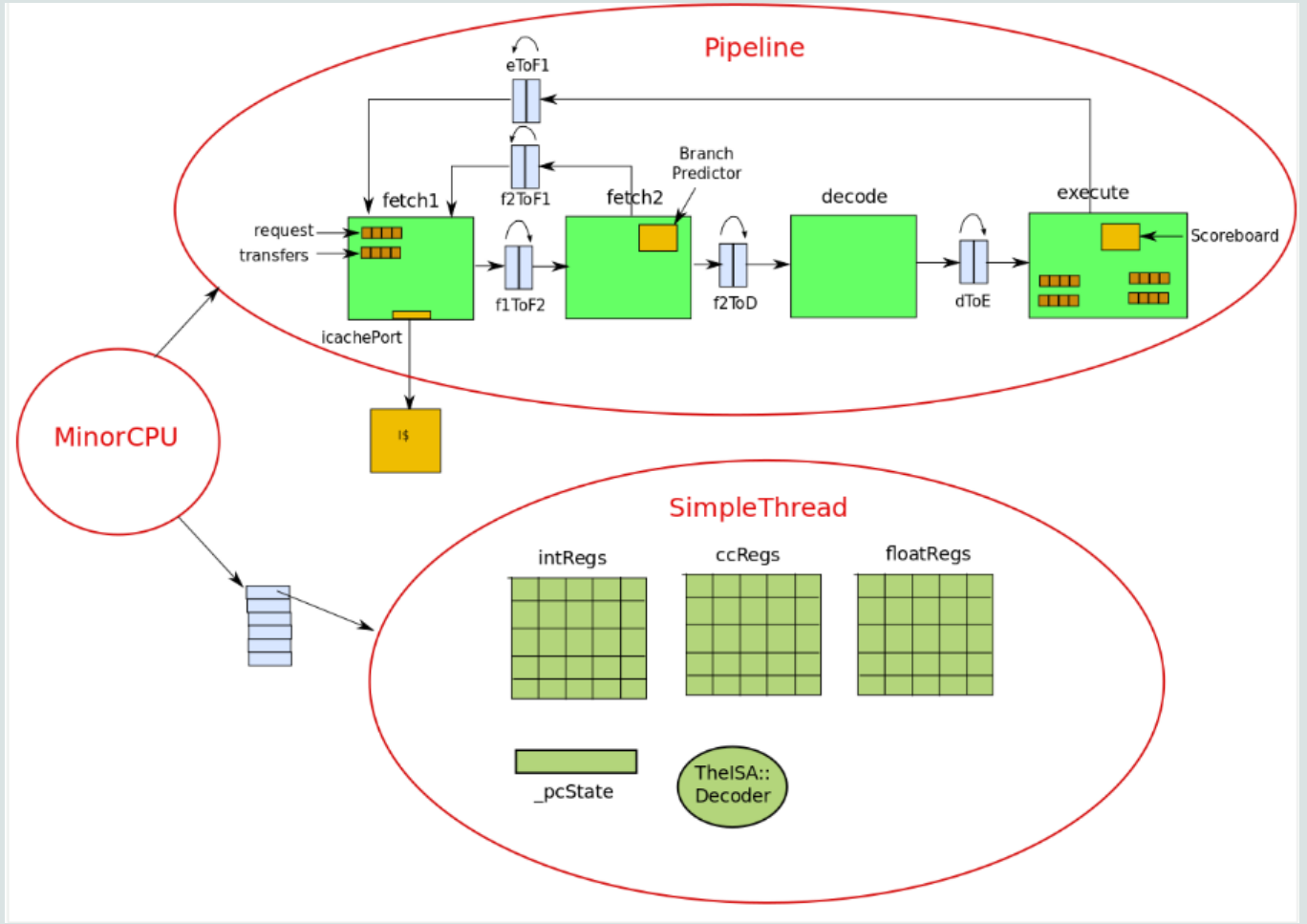
Inter-stage delay params

stage width params

```
decodeToFetchDelay = Param.Cycles(1, "Decode to fetch delay")
renameToFetchDelay = Param.Cycles(1 ,"Rename to fetch delay")
iewToFetchDelay = Param.Cycles(1, "Issue/Execute/Writeback to fetch "
                               "delay")
commitToFetchDelay = Param.Cycles(1, "Commit to fetch delay")
fetchWidth = Param.Unsigned(8, "Fetch width")
fetchBufferSize = Param.Unsigned(64, "Fetch buffer size in bytes")
fetchQueueSize = Param.Unsigned(32, "Fetch queue size in micro-ops "
                                "per-thread")

renameToDecodeDelay = Param.Cycles(1, "Rename to decode delay")
iewToDecodeDelay = Param.Cycles(1, "Issue/Execute/Writeback to decode "
                                "delay")
commitToDecodeDelay = Param.Cycles(1, "Commit to decode delay")
fetchToDecodeDelay = Param.Cycles(1, "Fetch to decode delay")
decodeWidth = Param.Unsigned(8, "Decode width")
```

configurable buffer sizes

MinorCPU



[1]

[1] https://nitish2112.github.io/post/gem5-minor-cpu/

O3CPU Model Parameters (very configurable)

src/cpu/o3/BaseO3CPU.py

LQ/SQ Buffers

```
LQEntries = Param.Unsigned(32, "Number of load queue entries")
SQEntries = Param.Unsigned(32, "Number of store queue entries")
LSQDepCheckShift = Param.Unsigned(4,
        "Number of places to shift addr before check")
LSQCheckLoads = Param.Bool(True,
    "Should dependency violations be checked for "
    "loads & stores or just stores")
store_set_clear_period = Param.Unsigned(250000,
        "Number of load/store insts before the dep predictor "
        "should be invalidated")
LFSTSize = Param.Unsigned(1024, "Last fetched store table size")
SSITSize = Param.Unsigned(1024, "Store set ID table size")

numRobs = Param.Unsigned(1, "Number of Reorder Buffers");

numPhysIntRegs = Param.Unsigned(256,
        "Number of physical integer registers")
numPhysFloatRegs = Param.Unsigned(256, "Number of physical floating point "
                                       "registers")
numPhysVecRegs = Param.Unsigned(256, "Number of physical vector "
                                     "registers")
numPhysVecPredRegs = Param.Unsigned(32, "Number of physical predicate "
                                        "registers")
# most ISAs don't use condition-code regs, so default is 0
numPhysCCRegs = Param.Unsigned(0, "Number of physical cc registers")
numIQEntries = Param.Unsigned(64, "Number of instruction queue entries")
numROBEntries = Param.Unsigned(192, "Number of reorder buffer entries")
```

Number of Physical regs.

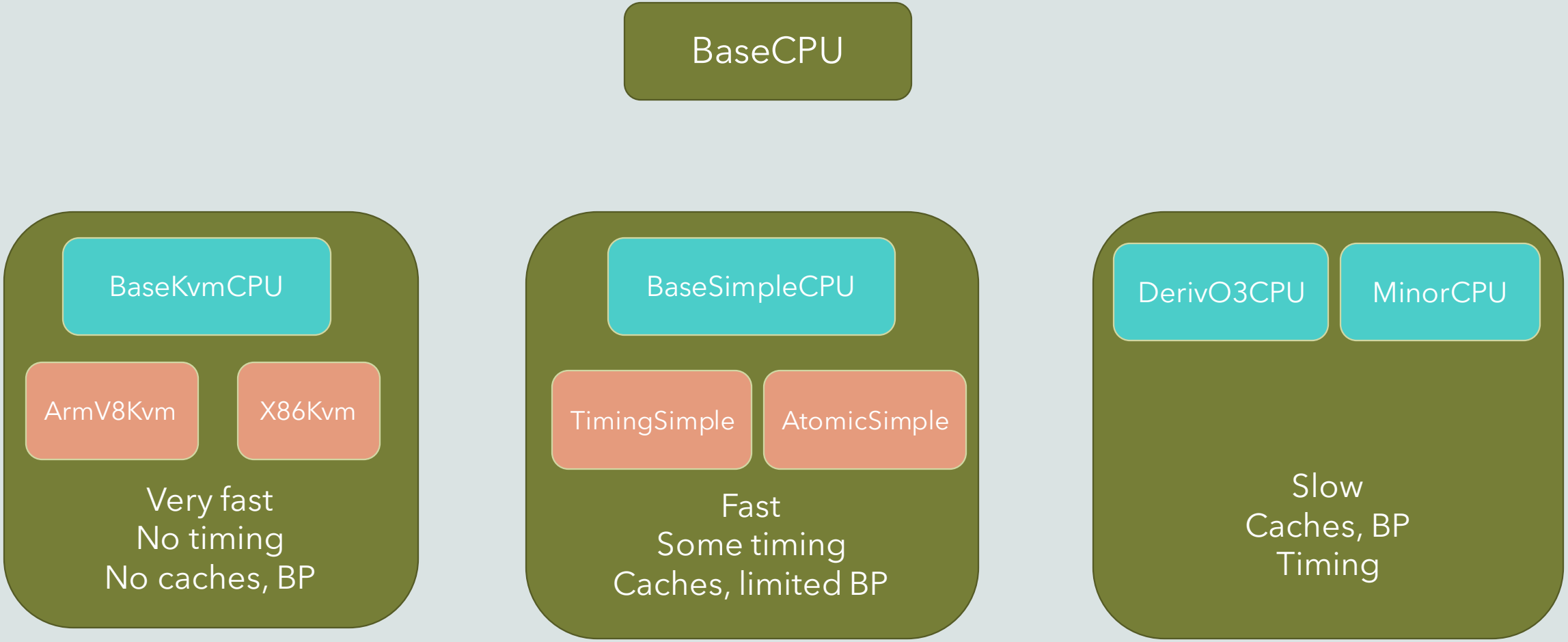
Reservation station size

ROB size

KvmCPU

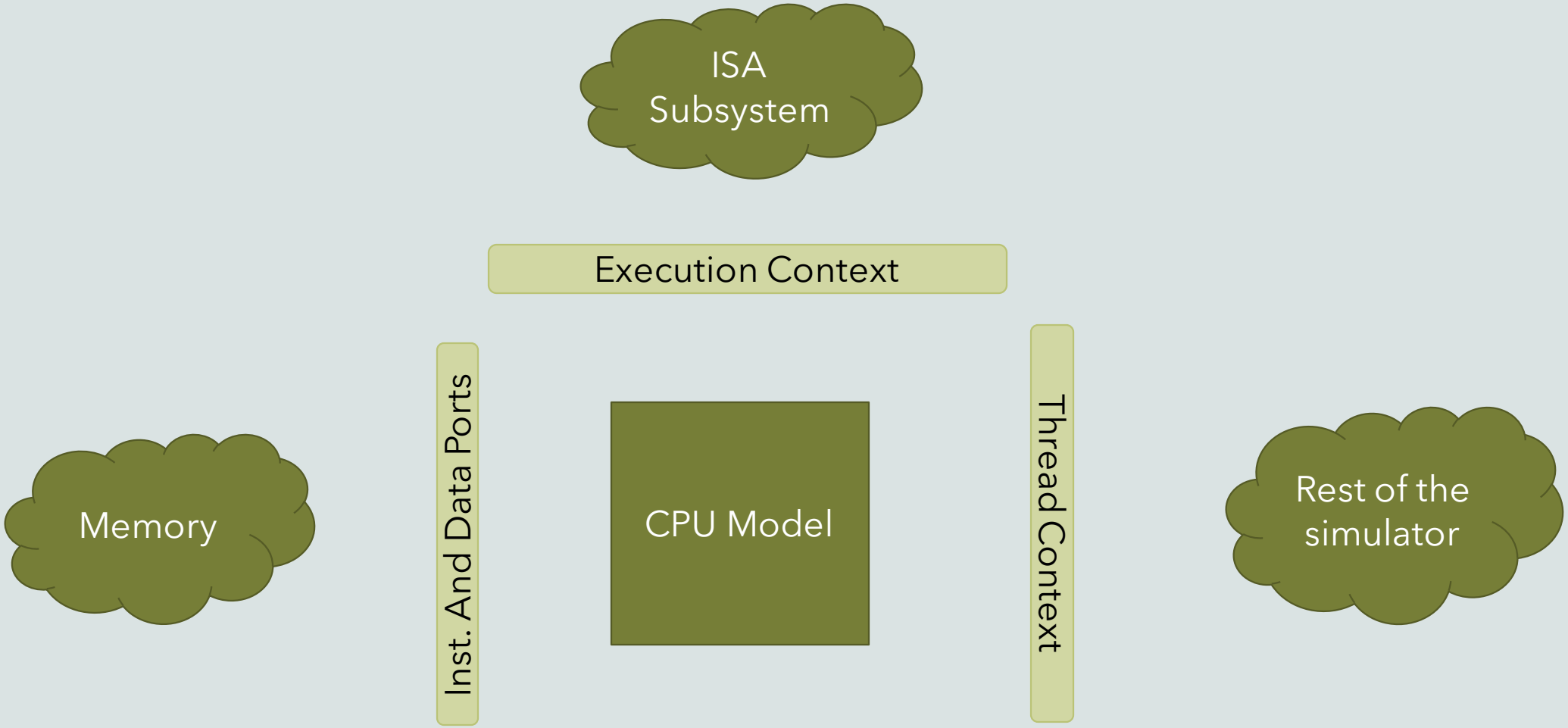
- KVM – Kernel-based virtual machine
- Used for native execution on x86 and ARM host platforms
- guest and the host need to have the same ISA
- Very useful for functional tests and fast-forwarding

Summary of gem5 CPU Models



gem5

Interaction of CPU model with other parts of gem5



Outline

CPU models in gem5

- AtomicSimpleCPU, TimingSimpleCPU, O3CPU, MinorCPU, KvmCPU

Using the CPU models

- Set-up a simple system with two cache sizes and three CPU models

Look at the gem5 generated statistics

- To understand differences among CPU models

Let's use these CPU models!

Material to use

gem5-bootcamp-env/materials/using-gem5/05-cpu-models/

- cpu-models.py
- IntMM/
- finished-material/

Let's configure a simple system with Atomic CPU

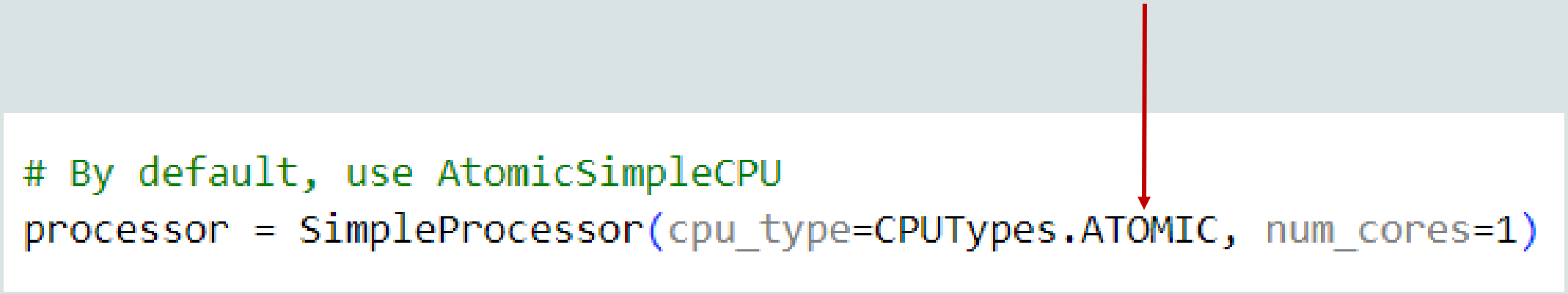
```
from gem5.components.boards.simple_board import SimpleBoard
from gem5.components.cachehierarchies.classic.private_l1_cache_hierarchy import PrivateL1CacheHierarchy
from gem5.components.memory.single_channel import SingleChannelDDR3_1600
from gem5.components.processors.simple_processor import SimpleProcessor
from gem5.components.processors.cpu_types import CPUTypes
from gem5.resources.resource import CustomResource
from gem5.simulate.simulator import Simulator

# A simple script (similar to the hello-world script from the stdlib tutorial)
# to test with different CPU models
# We will run a simple application with AtomicSimpleCPU, TimingSimpleCPU, and
# O3CPU using two different cache sizes
```

```
# O3CPU using two different cache sizes

cache_hierarchy = PrivateL1CacheHierarchy(l1d_size="32KiB", l1i_size="32KiB")
memory = SingleChannelDDR3_1600("1GiB")

# By default, use AtomicSimpleCPU
processor = SimpleProcessor(cpu_type=CPUTypes.ATOMIC, num_cores=1)

# Uncomment one of the following lines to use TimingSimpleCPU, or O3CPU
#processor = SimpleProcessor(cpu_type=CPUTypes.TIMING, num_cores=1)
#processor = SimpleProcessor(cpu_type=CPUTypes.O3, num_cores=1)


#Add them to the board.
board = SimpleBoard(
    clk_freq="3GHz",
    processor=processor,
    memory=memory,
    cache_hierarchy=cache_hierarchy,
)

# Set the workload to be a modified version of IntMM (benchmark from llvm tests).
# The matrix size is modfied to reduce the simulation time.
binary = CustomResource("materials/using-gem5/05-cpu-models/IntMM/IntMM")
board.set_se_binary_workload(binary)

# Setup the Simulator and run the simulation.
simulator = Simulator(board=board)
simulator.run()
```

Building the test binary

```
> cd /workspaces/gem5-bootcamp-env/materials/using-gem5/05-cpu-models/IntMM

> make
```

Change the CPU model to timing, and O3



Let's run it!

```
> gem5-x86 --outdir=atomic-normal-cache materials/using-gem5/05-cpu-models/cpu-models.py
```

Let's run again!

Don't forget to change the name of the output directory to (timing or o3)


```
> gem5-x86 --outdir=timing-normal-cache materials/using-gem5/05-cpu-models/cpu-models.py
```

Let's use the small cache with all CPU models

Set the output directory depending on the CPU model you are using


```
> gem5-x86 --outdir=atomic-small-cache materials/using-gem5/05-cpu-models/cpu-models.py
```

Outline

CPU models in gem5

- AtomicSimpleCPU, TimingSimpleCPU, O3CPU, MinorCPU, KvmCPU

Using the CPU models

- Set-up a simple system with two cache sizes and three CPU models

Look at the gem5 generated statistics

- To understand differences among CPU models

Now, let's change the cache size!

```
cache_hierarchy = PrivateL1CacheHierarchy(l1d_size="32KiB", l1i_size="32KiB")
memory = SingleChannelDDR3_1600("1GiB")
```



```
# let's use a very small cache, to see the impact on modeling of memory accesses
cache_hierarchy = PrivateL1CacheHierarchy(l1d_size="1KiB", l1i_size="1KiB")
memory = SingleChannelDDR3_1600("1GiB")
```

gem5

Time to look at the generated stats.!

gem5

Look at the number of operations

grep –ri "simOps"

Look at the number of operations

grep -ri "simOps"

```
stats-timing.txt:simOps                                    2308506
stats-o3_cpu.txt:simOps                                    2308506
stats-atomic.txt:simOps                                    2308506
```

Compare execution cycles

grep –ri "numCycles"

Compare execution cycles

Normal cache size

```
stats-timing.txt:board.processor.cores.core.numCycles          3153969
stats-o3_cpu.txt:board.processor.cores.core.numCycles          1254910
stats-atomic.txt:board.processor.cores.core.numCycles          2879828
```

gem5

Compare execution cycles

Normal cache size

```
stats-timing.txt:board.processor.cores.core.numCycles      3153969
stats-o3_cpu.txt:board.processor.cores.core.numCycles      1254910
stats-atomic.txt:board.processor.cores.core.numCycles      2879828
```

Small cache size

```
stats-timing.txt:board.processor.cores.core.numCycles      5398034
stats-o3_cpu.txt:board.processor.cores.core.numCycles      2999974
stats-atomic.txt:board.processor.cores.core.numCycles      2879828
```

Compare execution cycles

Normal cache size

```
stats-timing.txt:board.processor.cores.core.numCycles      3153969
stats-o3_cpu.txt:board.processor.cores.core.numCycles      1254910
stats-atomic.txt:board.processor.cores.core.numCycles      2879828
```

Compare execution cycles

Normal cache size

```
stats-timing.txt:board.processor.cores.core.numCycles      3153969
stats-o3_cpu.txt:board.processor.cores.core.numCycles      1254910
stats-atomic.txt:board.processor.cores.core.numCycles      2879828
```

Small cache size

```
stats-timing.txt:board.processor.cores.core.numCycles      5398034
stats-o3_cpu.txt:board.processor.cores.core.numCycles      2999974
stats-atomic.txt:board.processor.cores.core.numCycles      2879828
```

Compare execution cycles

Normal cache size

```
stats-timing.txt:board.processor.cores.core.numCycles          3153969
stats-o3_cpu.txt:board.processor.cores.core.numCycles          1254910
stats-atomic.txt:board.processor.cores.core.numCycles          2879828
```

Small cache size

```
stats-timing.txt:board.processor.cores.core.numCycles          5398034
stats-o3_cpu.txt:board.processor.cores.core.numCycles          2999974
stats-atomic.txt:board.processor.cores.core.numCycles          2879828
```

Other statistics you might want to look at!

Host time (time taken by gem5 to run your simulation)

hostSeconds

For O3CPU

branchMispredicts

commitSquashedInsts

Compare execution cycles

Normal cache size

```
stats-timing.txt:board.processor.cores.core.numCycles      3153969
stats-o3_cpu.txt:board.processor.cores.core.numCycles      1254910
stats-atomic.txt:board.processor.cores.core.numCycles      2879828
```

Small cache size

```
stats-timing.txt:board.processor.cores.core.numCycles      5398034
stats-o3_cpu.txt:board.processor.cores.core.numCycles      2999974
stats-atomic.txt:board.processor.cores.core.numCycles      2879828
```