CS570 "Advanced Computer Architecture"
Homework 6

Submission:
Due by end of the day of 03/24/2020
This is an individual assignment!
Total points 100 - Late penalty: 10% penalty for each day late
Please upload your assignment on Blackboard with the following name:
CS570_SectionNumber_LastName_FirstName_HW6.zip.
Please do NOT email your assignment to the instructor and/or TA!

In this assignment, you run gem5 with some configurations such as CPU configurations and cache configurations. We recommend a **64-bit Linux machine** or **virtual machine** to do this assignment. If you use a virtual machine, you may increase the amount of memory allocated to it.

1. Background

Deep memory hierarchy can be utilized to mitigate the speed gap between the processor and the memory. Temporal and spatial locality ensures the practicality of deep memory hierarchy, which gives the allusion of a large, fast memory being presented to the processor. Memory hierarchy design becomes more crucial with recent multi-core processors. In this assignment, you are asked to implement the deep cache hierarchy on X86 architecture based on the gem5 simulator and explored the effect of deep cache hierarchy's design on the multicore processor's performance.

2. Supplied Benchmark Program

BFS: computes a breadth-first search problem. This is taken from the Problem Based Benchmark Suite. Source code for this benchmark, along with utilities for generating graph data is in the breadthFirstSearch directory.
We supply some example graphs in the inputs directory. Our suggested command-line for this program is

```
./BFS path/to/RL3k.graph
```

This program was selected because it should have poor data cache locality.
Please download from https://github.com/Xiaoyang-Lu/CS570-benchmark.git

3. Task 1 – Add shared L3 cache in gem5

In this part, you should add a shared L3 cache in gem5 system simulator and modify the gem5 code so that each core has a private L2 cache.

The default hardware configurations are as follows:

- CPU mode: O3 mode
- The number of CPU cores: 1, 2, 4

- CPU frequency: 2GHz
- Caches: with three level caches which are L1 cache and L2 cache and L3 cache
  2-way L1 dcache with the size of 32KB per core
  2-way L1 icache with the size of 32KB per core
  4-way L2 cache with the size of 128KB per core
  16-way shared L3 cache with the size of 1MB
  The cacheline size of these caches are 64 Byte
- Memory mode: Simple Memory
- Memory size: 8GB

By default, gem5 does not have the L3 cache support. You need to add the L3 support by yourself. Below are the steps of adding L3 cache in gem5:
**Due to different versions of gem5, the following code may not be completely applicable to your version, but the logic should be the same, so please read the code carefully when operating.*

1. Add L3 cache option in configs/common/Options.py:

```
parser.add_option("--l3cache", action="store_true")
parser.add_option("--l3_size", type="string", default="16MB")
parser.add_option("--l3_assoc", type="int", default=16)
```

2. Add L3 cache class in configs/common/Caches.py:

```
class L3Cache(Cache):
   assoc = 64
   tag_latency = 32
   data_latency = 32
   response_latency = 32
   mshrs = 32
   tgts_per_mshr = 24
   write_buffers = 16
```

3. Define a L3XBar class in /src/mem/XBar.py:

```
class L3XBar(CoherentXBar):
   width = 32
   frontend_latency = 1
   forward_latency = 0
   response_latency = 1
   snoop_response_latency = 1
   snoop_filter = SnoopFilter(lookup_latency = 0)
```

4. import L3XBar from XBar in /src/cpu/BaseCPU.py:

```
from XBar import L3XBar
```

5. Define addThreeLevelCacheHierarchy function in /src/cpu/BaseCPU.py:

```
def addThreeLevelCacheHierarchy(self, ic, dc, l2c, iwc = None, dwc = None):
   self.addPrivateSplitL2Caches(ic, dc, iwc, dwc)
   self.toL3Bus = L3XBar()
   self.connectCachedPorts(self.toL3Bus)
   self.l3cache = l3c
   self.toL3Bus.master = self.l3cache.cpu_side
   self._cached_ports = ['l3cache.mem_side']
```

6. Assign L3Cache to l3_cache_class in /configs/common/CacheConfig.py:

```
   if options.cpu_type == "O3_ARM_v7a_3":
      try:
         from cores.arm.O3_ARM_v7a import *
      except:
         print "O3_ARM_v7a_3 is unavailable. Did you compile the O3 model?"
         sys.exit(1)

      dcache_class, icache_class, l2_cache_class, l3_cache_class, walk_cache_class = \
         O3_ARM_v7a_DCache, O3_ARM_v7a_ICache, O3_ARM_v7aL2, \
         O3_ARM_v7aWalkCache
   else:
      dcache_class, icache_class, l2_cache_class, l3_cache_class, walk_cache_class = \
         L1_DCache, L1_ICache, L2Cache, L3Cache, None

      if buildEnv['TARGET_ISA'] == 'x86':
         walk_cache_class = PageTableWalkerCache
```

7. Please modify the code in /configs/common/CacheConfig.py, so that each core has an L2 private cache. Finally, please connect private L2s, shared L3 and mem buses in CacheConfig.py.

Every time you make a change to the source files, you need to compile to see the result.

For visualizing the configuration used for the memory hierarchy (i.e. how the memory objects are connected to the ports). Please Install pydot, and then rebuild gem5 and run the simulation. You can see config.dot.pdf file under the m5out folder.

After completing the compilation, run the test HelloWorld program, with Syscall Emulation (SE) system mode in a 2-core configuration:

```
build/X86/gem5.opt ./configs/example/se.py --caches --l1d_size=32kB --l1d_assoc=2 --
l1i_size=32kB --l1i_assoc=2 --l2cache --l2_size=128kB --l2_assoc=4 --l3cache --l3_size=1MB --
```

```
l3_assoc=16 --cacheline_size=64 --cpu-type=DerivO3CPU --mem-type=SimpleMemory --
mem-size=8192MB -c './tests/test-progs/hello/bin/x86/linux/hello
; ./tests/test-progs/hello/bin/x86/linux/hello' --cpu-clock=2GHz -n 2
```

Now if your compilation was successful, you should be able to check the L3 cache being used in the config.dot.pdf file. Please present the config.dot.pdf file to prove the correctness of your design.

4. Task 2 – Performance Evaluation

You need to run BFS benchmark with RL3k.graph that you downloaded in Part 2. Then design the performance test plot for a shared L3 cache for the capacity of the L3 cache and the number of cores (the number of individual L2 cache). Please change the core number with the value of 1->2->4; change the size of L3 cache with the value of 512KB->1MB->2MB. You need to try a total of 9 (3*3) configurations. Please explore the performance difference (average IPC value across all cores) between different configurations.

You should explain the reasoning about your result for each configuration (utilize the L3 miss rate for auxiliary analysis). Please present a graph to show the relationship between the configurations and average IPC value across all cores and present another graph to show the relationship between the configurations and L3 miss rate.

You can use the following command to run gem5 with Syscall Emulation (SE) system mode:

```
build/X86/gem5.opt ./configs/example/se.py --caches --l1d_size=32kB --l1d_assoc=2 --
l1i_size=32kB --l1i_assoc=2 --l2cache --l2_size=128kB --l2_assoc=4 --l3cache --l3_size=512kB -
-l3_assoc=16 --cacheline_size=64 --cpu-type=DerivO3CPU --mem-type=SimpleMemory --
mem-size=8192MB -c 'PATHTO/BFS; PATHTO/BFS' -o ' PATHTO/RL3k.graph; PATHTO
/RL3k.graph' --cpu-clock=2GHz -n 2
```

If you do not know that how many available configuration options your gem5 has, you can use the command "./build/X86/gem5.opt configs/example/se.py --help" to check these configuration options.

5. What to turn in

Your homework should be submitted as a zip file and should include:

1. The config.dot.pdf file which is generated by gem5 after running gem5 in 2-core configuration.
2. Please provide the changes you made to the source code in order to add the L3 cache in Task 1. Please only submit your edited CacheConfig.py.
3. A written report of the assignment in PDF format. This is your chance to explain your steps of running gem5 with these configurations, the explanation of your results and the graphs about different configurations and performance are needed.