3.	Fachgebiet Architektur eingebetteter Systeme **Advanced Computer Architectures**	SoSe2018

Assignment start: 05.06.2018 Submission deadline: 25.06.2018

Assignment 3 - Sim Cache 30 Points

It is a very common exercise for computer architects to evaluate the design space of different architectures. In this regard, simulators play a significant role in investigating different design choices and making early decisions. Every computer architect is expected to be well versed with the usage and modification of simulators, to evaluate different alternative designs. In the pursuit of mastering these skills, we will use and modify an existing cache simulator in this exercise.

The objective of the third assignment is to explore the design space of cache and memory hierarchy organization using the *sim-cache* simulator, which is a part of the SimpleScalar simulator. You will investigate the changes in the miss rate due to different configurable parameters and different choices to organize multi-level caches. Documentation about the SimpleScalar simulator is available at `http://www.simplescalar.com/docs/users_guide_v2.pdf`. SimpleScalar contains different simulators. In this assignment we only use the *sim-cache* version of the simulator. The simulator only runs on Linux (x86, 64-bit).

The simulator and the pre-compiled benchmarks can be found at the following location and also on the ISIS course page.
`/afs/tu-berlin.de/units/Fak_IV/aes/aca/simCache`
Following is the list of benchmarks that we will use:

- *Fibonacci* - calculates the 20th instance of the Fibonacci sequence
- *Matmul* - matrix multiplication of size 50x50
- *Pi* - estimates the pi number
- *Whetstone* - a double precision benchmark
- *Memcopy* - a memory benchmark

1. (5 Points) Investigate the effect of **cache size and block size**. Please use the following configurations:

   - Vary the number of L1 data cache lines: 1K, 2K, 4K, 8K, 16K, 32K, 64K, 128K for each block size of 8B, 16B, 32B, 64B.
   - Configure the cache to be direct mapped.
   - Configure the cache to use Least-Recently-Used (LRU) replacement policy.
   - Default size for L1 instruction cache and default unified L2 cache.

   Perform the experiments for each benchmark and plot the miss rate results in a line graph. Compare the change in miss rate within a single benchmark as well as across benchmarks.

2. (5 Points) Investigate the effect of **associativity** on the miss rate. Please use the following configurations:

   - A split L1 cache and unified L2 cache.
   - Set L1 data cache size to 8KB and default size for L1 instruction cache.
   - Set L2 cache size to 256KB.

- Vary the associativity from 1-way, 2-way, 4-way, 8-way to 16-way. Please also vary the number of sets to keep the total cache size constant.
- Set block size to 32B.
- Set the cache configuration to use LRU replacement policy.

Perform the experiments for each benchmark and plot the miss rate results in a line graph. Again compare the change in miss rate within a single benchmark as well as across benchmarks. You have to at least comment on the effect of increasing associativity on the miss rate.

3. (5 Points) Investigate the effect of **unified vs split cache**. Perform experiments with both unified and split L1 and L2 caches for each benchmark. Please use the following configurations:
   - Set the L1 cache size to 16KB. When configured as split cache, use 8KB for for L1 instruction cache and remaining as L1 data cache.
   - Set L2 cache size to 256KB. When configured as split L2 cache, use 64KB for L2 instruction cache and remaining for L2 data cache.
   - Set the associativity to 4-way and block size to 32B.
   - Set the cache configuration to use LRU replacement policy.

   Perform the experiments for each benchmark and plot the miss rate results in a bar graph. What L1 and L2 organization would you recommend?

4. (5 Points) Investigate the effect of **block replacement policy**. Perform your experiments with the three replacement policies (LRU, FIFO, and Random). Please use the following configurations:
   - A unified L1 cache and no L2 cache for this exercise.
   - Assume L1 cache size of 8KB with 16-way associativity and block size of 32B.

   Compare and plot the data cache miss rates.

5. (10 Points) In the last exercise you are supposed to modify the cache simulator to implement a not-recently-used (NRU) cache replacement policy. Compare the performance of the NRU replacement policy (described below) with a pure LRU replacement policy for a unified L1 cache. Use the configurations that are used to investigate the block replacement policy. The SimpleScalar cache simulator already comes with an implementation of the LRU policy. You are required to implement the NRU policy inside sim-cache. Please compare the miss rates of LRU and NRU.

   What will be the reasons for using NRU instead of an LRU?

   NRU: The OpenSparc T1/T2 processors implement a not-recently-used (NRU) replacement policy for the unified second level cache. This section describes a (simplified) NRU policy that you are required to implement inside the SimpleScalar sim-cache simulator. The NRU replacement policy is based on a used-bit scheme. Each cache line has a used bit associated with it. Initially, all bits are reset (i.e., all used bits in the cache are zero). The used bit is set each time a cache line is accessed or when initially fetched/filled from memory. If setting the used bit for the current line causes all used bits within the corresponding cache set to be set to one, all other bits are reset. This ensures that at any time there is at least one cache line within a set that has its used bit clear. On a cache miss, a cache line with its used bit clear is chosen to be replaced. The L2 cache has a single rotating replacement pointer, which is the "starting point" to find the way to replace. On a miss, the L2 looks for the first line within the set with its used bit clear, starting with the way pointed at by the replacement pointer. The replacement pointer is then rotated forward one way. Using the same replacement pointer for all sets of the L2 cache makes the replacement be more random than if each set had its own replacement pointer. Initially, the replacement pointer points to the first way (i.e., way zero).

Submission: Please submit a report covering experimental results and your observations for items 1-5. You also need to submit the modified version of sim-cache that supports NRU.

Note: whereever nothing is explicitly specified, please use the default configuration.