**Mahindra University, Spring 2021**

**EE318 Computer Architecture and Design**

**Cache Simulator Programming Assignment**

**Submission Deadline: 09/05/2021**

**Section I: Introduction**

In this assignment, you are required to implement a **single level cache simulator** and analyze the performance of various cache architectures using real-world program traces. For this purpose, you will write programs **(C/C++/Python)** to read in the given memory traces and simulate different cache architectures and determine the performance and overheads.

Information about the traces are given in Section II. The cache architectures you required to implement are listed in Section III. Some of the architectures are mandatory to implement for all students. The results to be reported are given in Section IV. Apart from this, students have the liberty to explore research papers to implement cache architectures and replacement algorithms to take the performance as close to an ideal cache as possible i.e. 100% hit-rate. A leaderboard will be created and shared to keep track of the best algorithm that the class has designed and implemented. This is described in Section V. *This is optional*. Section VI describes the submission procedure.

**Section II: About the Traces**

The **traces** you have been given are generated from runs of a subset of programs from the **MiBench** embedded benchmark suite (http://vhosts.eecs.umich.edu/mibench/). The link also contains a characterization reference paper which you can peruse for sanity checks for your cache simulator. They contain the complete instruction and data memory access trace of the programs. Each line of the trace file has only one field – the memory address accessed in 64-bit HEX format. Below are the first four lines of a trace file:

0x00007FFFD9C0EDB0

0x00007FFFD9C0EC98

0x00007FFFD9C0ECAC

0x00007FFFD9C0ECB0

You can also generate traces of the remaining benchmarks and see how they perform for each cache architecture.

**Section III: The Cache Simulator**

Any cache architecture has 5 primary configuration parameters

1. **CSIZE:** Cache Size (data alone)
2. **BSIZE:** Block Size,
3. **ASSOC:** Associativity,
4. **RPOL:** Replacement policy (for associative caches),
5. **WPOL:** Write policy

Your cache simulator should accept the parameters listed above and determine the required performance metrics by simulating each line of the trace file sequentially. For this, you will have to maintain a data structure for the cache and manipulate it according to each memory access. Simulate each of the following architecture configurations and report the performance metrics in the required formats as given at the end of this section.

1. **CSIZE:** vary from 256 bytes to 4MB in convenient strides
2. **BSIZE:** vary from 4 bytes to 512 bytes
3. **ASSOC:** Direct Mapped, 2-way, 4-way, 8-way, 16-way
4. **RPOL:** LRU, NFU, CLOCK, RANDOM (only for set-associative caches)
5. **WPOL:** Write Through, Write Allocate

**\*PLEASE NOTE THAT COME OF THE CONFIGURATIONS ARE NOT PRACTICALLY POSSIBLE e.g. BSIZE > CSIZE, or (Num. of Ways \* Num. of Sets \* BSIZE) > CSIZE. Prune these configurations out before simulating.**

**Section IV: Performance Metrics to be reported**

1. For each cache configuration and benchmark combination (and all benchmarks cumulatively):
2. Given ASSOC and CSIZE, vary BSIZE and plot miss-rate (Total, Cold, Conflict, Capacity). Do the same for different combinations of ASSOC and CSIZE
3. Given ASSOC and BSIZE, vary CSIZE and plot miss-rate (Total, Cold, Conflict, Capacity). Do the same for different combinations of ASSOC and BSIZE
4. Given CSIZE and BSIZE, vary ASSOC and plot miss-rate (Total, Cold, Conflict, Capacity). Do the same for different combinations of BSIZE and CSIZE

You can also plot a 3D graph varying all 3 parameters.

1. Also, plot the Storage Overhead for each cache architecture (does not vary with benchmark).
2. The best cache architecture for each of the given benchmark memory traces.

**Section V: Cache Design Competition (optional)**

Implement your own cache architecture (mainly varying the replacement policy, or implementing pre-fetching of some kind) to provide the highest performance possible for the all of the memory traces given. Submit this as a separate program in the final submission.

**Section VI: Submission Format**

Your submission should be a compressed file (<10 MB) with the following contents:

1. Source code of the configurable cache simulator and the customized cache design (Section V)
2. Readme file on how to run the code.
3. Document listing the performance and analysis of all the cache architecture iterations given. The required plots should also be computed and presented. The document should be in the same format as this assignment sheet.

**The compressed file should be uploaded at before 07/04/2021 midnight IST.**

**PLEASE NOTE:**

* **The code should be properly commented.**
* **I should not be required to edit anything in the code in order for me to run it.**
* **The trace files have to be taken as command line parameters.**