Computer Architecture - CS2323. Autumn 2023

Lab-6 (Cache Miss Simulator)

In this assignment, we will simulate a cache to identify which accesses would cause a cache hit and which accesses would cause a cache miss. The program should allow customization for various cache parameters as follows:

a. Total size of the cache (specified in Bytes)
b. Size of each cache block (specified in Bytes)
c. Associativity (1: Direct mapped, 0: Fully associative, any other number: set associative) - upto 16
d. Replacement policy (FIFO, LRU, RANDOM)
e. Write policy (WB: WriteBack, WT: WriteThrough). Assume No-Allocate for WT.

The cache configuration is provided as an input file (cache.config) in the following format:

SIZE_OF_CACHE (number)
BLOCK_SIZE (number)
ASSOCIATIVITY (number)
REPLACEMENT_POLICY (FIFO or LRU or RANDOM)
WRITEBACK_POLICY (WB or WT)

The access sequence is provided in another file (cache.access) in the following format:

Mode: Address
Mode: Address
…

Mode refers to access mode: R: Read or W: Write
Address refers to the memory address that is accessed

For simplicity (if it helps), assume the following bounds and steps:

a. Cache size: max. 1 MB, always power of 2
b. Block size: max. 64 Bytes, always power of 2
c. Associativity: max. 16, always power of 2
d. Input address: always 32-bits, always power of 2

Problem Statement: Implement a C/C++ program to simulate cache behavior as described above. For each access, your program should output the index/set for the given address, whether it is a hit or miss, and the TAG being stored in the cache.

You could approach the problem in multiple sub-parts, as mentioned below, with partial weightages for each part. **There will be no partial marking within each part, so verify your implementation properly.**

1. **Part-1:** Only support read access modeling, for a direct-mapped cache, with FIFO replacement policy (30%)
2. **Part-2:** Part-1 + LRU and RANDOM replacement policies (+20%)
3. **Part-3:** Part-2 + caches with associativity (+25%)
4. **Part-4:** Part-3 + write access modeling (+25%)

Example config file:

```
32168
16
8
LRU
WT
```

Example input file:

```
R: 0x20203302
R: 0x20202011
W: 0x20203302
```

Example output (the values shown are just representative, not correct):

```
Address: 0x20203302, Set: 0x02, Hit, Tag: 0x202033
Address: 0x20202011, Set: 0x11, Miss, Tag: 0x202020
Address: 0x20203302, Set: 0x02, Miss, Tag: 0x202033
```

<u>Submission instructions:</u>

The assignment is to be done individually. Submit your code and instructions to compile/execute your code should be put in a separate README file. You should clearly indicate which parts (part-1, 2, 3, 4) are supported by your code. Prepare a short report on your coding approach and what all you did for testing your code to be correct. Submit a zip file containing the following:

1. Source files of code
2. README: A text file containing instructions to compile/execute your code
3. Testcases, if you tried out specific input files
4. report.pdf - a short report on your coding approach and what all you did for testing your code to be correct. If a proper report is not submitted, you will receive 10% lesser marks.
5. Submission deadline: 22 November 2023, 11:59 PM
   NO LATE SUBMISSION ALLOWED FOR THIS ASSIGNMENT.