Cache Simulator YANG MO A0091836X

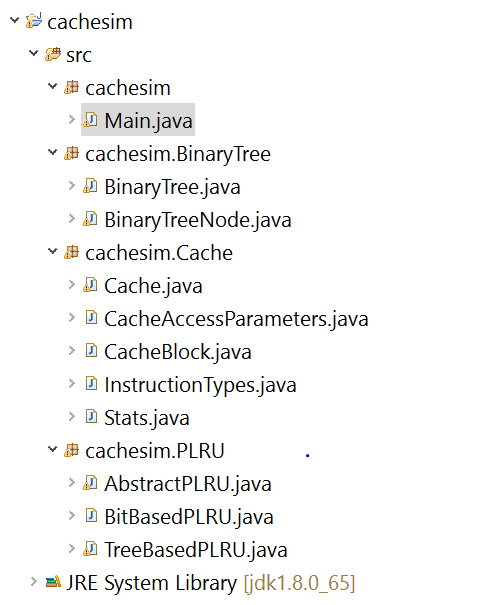
Preview

CPU cache is essential part of any modern processor design. Its purpose follows from speed limits of DRAM – there are several levels of memory with different ratios of the capacity to the speed between CPU and DRAM. This design allows to neutralize the difference in the speed of the CPU and DRAM.

Application Design

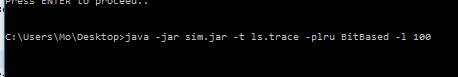
This application simulates operations in CPU cache of one of the possible configurations.  
Current cache parameters of the app as follows:

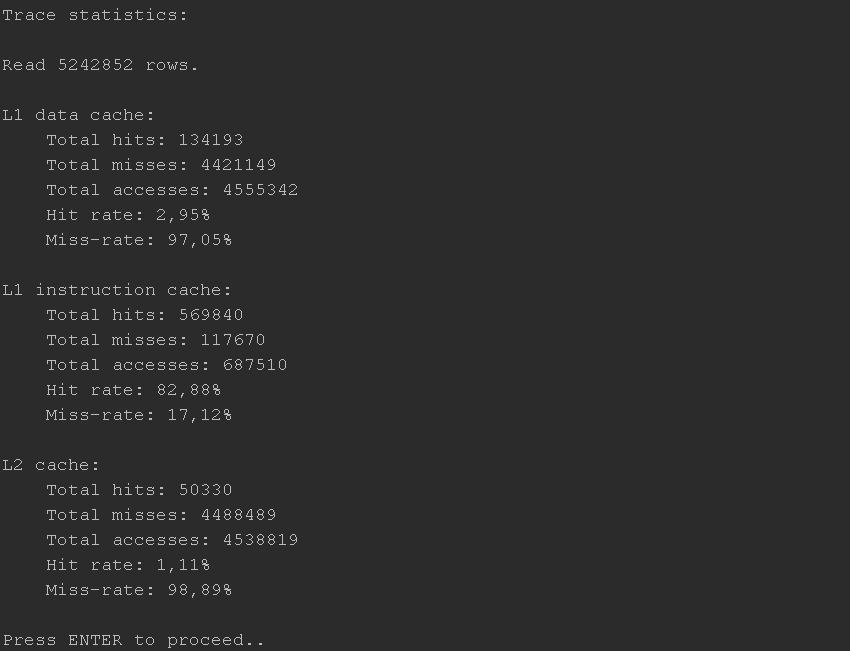
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cache level | Cache type | Capacity, Kbyte | Block size, Byte | Associativity, ways | Write policy |
| L1 | Instructions | 32 | 64 | 8 |  |
| L1 | Data | 32 | 64 | 8 | Write-back, write allocate |
| L2 | Unified | 32 | 64 | 8 | Write-through, write allocate |



The design is not complicated, but it allows flexible change of the cache configuration if necessary.

All cache logic completed with class *Cache* within package *Cache* This package also contains other service classes that are needed for *Cache*.  
 The *Cache* class could be instanced with different parameters as follows: size of instructions (bytes), cache capacity (Kbytes), cache block size (bytes), ways of associativity and PLRU type. Thus, creating instances of *Cache* with different parameters, we could simulate different configurations of CPU cache so.  
  
 Also, there is *PLRU* package. It contains classes implementing Pseudo LRU. The app allows create two type of PLRU – bit based and tree based. Reference to instances of both PLRU types could be passed as *Cache* class constructor parameters, thereby any *Cache* instance could be combinated with any PLRU (or it could be missed and cache will be simulated without PLRU).  
 *BinaryTree* package contains implementation of binary tree used by *TreeBasedPLRU* class.  
  
 CPU cache bus provided transport used by different levels of cache. Via this bus CPU organizes interactions with all caches. Implementation of bus behavior is modeled in *Main* class. So if bus logic needs to reorganize we could just change the *Main* logic.  
 In *Main* class instances of *Cache* and *PLRU* created with characteristics from table above. Also, victim cache instance created for L1 cache level . Further, *Main* class contains algorithm which instruction by instruction trace the source trace file.

Usage  
  
 Application requires mandatory command line option *–t trace\_path* where *trace\_path* is the full path to source trace file with instructions.  
 Also, there is two optional arguments:  
 *-l count\_limit*, where *count\_limit* is the number of reads limit.  
 *-plru plru\_type*, where *plru\_type* is could be *BitBased* or *TreeBased.* It changes the PLRU type.  
  
 

Let’s try to start the app with the following command line parameters: *-t ls.trace -plru TreeBased*The trace result output below: **

As we could see the result counts total hits and misses and accesses, hit and miss ratios for each used cache level.

Let’s run the app with other configuration – we will use *gz.trace* trace file (with 10 000 000 reads limit) and bit based PLRU: *-t gz.trace -plru BitBased -l 10000000*  
The trace result output below:  
