

cachessim

1.0

Generated by Doxygen 1.8.0

Tue Apr 17 2012 10:39:37



# Contents

<b>1</b>	<b> cachesim</b>	<b>1</b>
<b>2</b>	<b> Namespace Index</b>	<b>3</b>
2.1	Namespace List . . . . .	3
<b>3</b>	<b> Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b> File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b> Namespace Documentation</b>	<b>9</b>
5.1	util Namespace Reference . . . . .	9
5.1.1	Function Documentation . . . . .	9
5.1.1.1	ltrim . . . . .	9
5.1.1.2	operator<< . . . . .	9
5.1.1.3	padHex . . . . .	9
5.1.1.4	rtrim . . . . .	9
5.1.1.5	splitLine . . . . .	9
5.1.1.6	trim . . . . .	9
<b>6</b>	<b> Class Documentation</b>	<b>11</b>
6.1	Cache Class Reference . . . . .	11
6.1.1	Constructor & Destructor Documentation . . . . .	12
6.1.1.1	Cache . . . . .	12
6.1.1.2	~Cache . . . . .	12
6.1.2	Member Function Documentation . . . . .	12
6.1.2.1	exec . . . . .	12
6.1.2.2	findMatch . . . . .	12
6.1.2.3	getAssociativity . . . . .	13
6.1.2.4	getBlockSize . . . . .	13
6.1.2.5	getCacheSize . . . . .	13
6.1.2.6	getNumBlocks . . . . .	13

6.1.2.7	<a href="#">getNumSets</a>	13
6.1.2.8	<a href="#">init</a>	13
6.1.2.9	<a href="#">load</a>	13
6.1.2.10	<a href="#">loadFile</a>	13
6.1.2.11	<a href="#">loadFile</a>	13
6.1.2.12	<a href="#">popSlot</a>	13
6.1.2.13	<a href="#">store</a>	14
6.1.3	<a href="#">Member Data Documentation</a>	14
6.1.3.1	<a href="#">_associativity</a>	14
6.1.3.2	<a href="#">_blockSize</a>	14
6.1.3.3	<a href="#">_cacheSize</a>	14
6.1.3.4	<a href="#">_filename</a>	14
6.1.3.5	<a href="#">_fs</a>	14
6.1.3.6	<a href="#">_numBlocks</a>	14
6.1.3.7	<a href="#">_numSets</a>	14
6.1.3.8	<a href="#">mainMem</a>	14
6.1.3.9	<a href="#">OFF_BITMASK</a>	14
6.1.3.10	<a href="#">OFFWIDTH</a>	14
6.1.3.11	<a href="#">SET_BITMASK</a>	14
6.1.3.12	<a href="#">sets</a>	14
6.1.3.13	<a href="#">SETWIDTH</a>	14
6.1.3.14	<a href="#">TAG_BITMASK</a>	14
6.1.3.15	<a href="#">TAGWIDTH</a>	14
6.2	<a href="#">Cache::CacheResult Struct Reference</a>	14
6.2.1	<a href="#">Member Data Documentation</a>	15
6.2.1.1	<a href="#">hit</a>	15
6.2.1.2	<a href="#">value</a>	15
6.3	<a href="#">Cache::Slot Struct Reference</a>	15
6.3.1	<a href="#">Member Data Documentation</a>	15
6.3.1.1	<a href="#">d</a>	15
6.3.1.2	<a href="#">data</a>	15
6.3.1.3	<a href="#">fields</a>	15
6.3.1.4	<a href="#">V</a>	15
<b>7</b>	<b><a href="#">File Documentation</a></b>	<b>17</b>
7.1	<a href="#">cache.cc File Reference</a>	17
7.1.1	<a href="#">Function Documentation</a>	17
7.1.1.1	<a href="#">FromString</a>	17
7.2	<a href="#">cache.h File Reference</a>	17
7.2.1	<a href="#">Define Documentation</a>	18

7.2.1.1	BUSWIDTH	18
7.2.2	Function Documentation	18
7.2.2.1	FromString	18
7.3	cachesim.cc File Reference	18
7.3.1	Function Documentation	18
7.3.1.1	FromString	18
7.3.1.2	main	18
7.4	cachesim.h File Reference	18
7.4.1	Function Documentation	19
7.4.1.1	FromString	19
7.4.1.2	main	19
7.5	README.md File Reference	19
7.6	util.cc File Reference	19
7.7	util.h File Reference	19



# Chapter 1

## cachesim

A program designed to simulate a single-level, set-associative, LRU cache with a write-back and write-allocate write policy.

### Building

To build `cachesim`, simply run `make all`. Alternatively, if you need to generate full debug information, then use `make debug`. Use `g++ >= 4.3` due to use of a C++0x/C++11 header file.

### Running

After building, run with the following parameters:

```
./cachesim <tracefile> <cache-size> <n-way-associativity> <block-size>
```

### Tracefile

The tracefile should contain store and load instructions in the following format:

```
store <address in hex> <access size in bytes> <value in hex>
load <address in hex> <access size>
```

For example:

```
store 0x1234ab00 2 19ab
load 0x002a173f 4
```

### Contributors

Kevin Gao [kag45]

Oliver Fang [orf2]





## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">util</a> . . . . .	9
--------------------------------	---



## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Cache</a> . . . . .	11
<a href="#">Cache::CacheResult</a> . . . . .	14
<a href="#">Cache::Slot</a> . . . . .	15



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">cache.cc</a>	17
<a href="#">cache.h</a>	17
<a href="#">cachesim.cc</a>	18
<a href="#">cachesim.h</a>	18
<a href="#">README.md</a>	19
<a href="#">util.cc</a>	19
<a href="#">util.h</a>	19



## Chapter 5

# Namespace Documentation

### 5.1 util Namespace Reference

#### Functions

- `std::vector< std::string > splitLine` (const `std::string` *str*, const char *delim*)
- `std::ostream & operator<<` (std::ostream &*out*, const `Cache` &*c*)
- void `padHex` (std::ostream &*out*, char \**value*, const int *bytes*)
- `std::string & trim` (std::string &*s*)
- `std::string & ltrim` (std::string &*s*)
- `std::string & rtrim` (std::string &*s*)

#### 5.1.1 Function Documentation

5.1.1.1 `std::string & util::ltrim ( std::string & s )`

5.1.1.2 `std::ostream & util::operator<< ( std::ostream & out, const Cache & c )`

5.1.1.3 `void util::padHex ( std::ostream & out, char * value, const int bytes )`

5.1.1.4 `std::string & util::rtrim ( std::string & s )`

5.1.1.5 `std::vector< std::string > util::splitLine ( const std::string str, const char delim )`

5.1.1.6 `std::string & util::trim ( std::string & s )`





## Chapter 6

# Class Documentation

### 6.1 Cache Class Reference

```
#include <cache.h>
```

#### Classes

- struct [CacheResult](#)
- struct [Slot](#)

#### Public Member Functions

- [Cache](#) (const char \*f, const unsigned short cs, const unsigned short a, const unsigned short bs)
- [~Cache](#) ()
- const unsigned short [getCacheSize](#) () const
- const unsigned short [getAssociativity](#) () const
- const unsigned short [getBlockSize](#) () const
- const unsigned short [getNumBlocks](#) () const
- const unsigned short [getNumSets](#) () const
- void [loadFile](#) ()
- const bool [loadFile](#) (const char \*f)
- void [exec](#) ()

#### Private Member Functions

- void [init](#) ()
- [CacheResult](#) [store](#) (unsigned int address, unsigned short accessSize, char \*value)
- [CacheResult](#) [load](#) (unsigned int address, unsigned short accessSize)
- void [popSlot](#) (std::list< [Slot](#) > &s, std::list< [Slot](#) >::iterator &it)
- std::list< [Slot](#) >::iterator [findMatch](#) (std::list< [Slot](#) > &s, const uint32\_t address, [CacheResult](#) &cr, const unsigned short accessSize)

#### Private Attributes

- const char \* [\\_filename](#)
- std::ifstream [\\_fs](#)
- const unsigned short [\\_cacheSize](#)
- const unsigned short [\\_associativity](#)

- const unsigned short [\\_blockSize](#)
- const unsigned short [\\_numBlocks](#)
- const unsigned short [\\_numSets](#)
- const unsigned short [OFFWIDTH](#)
- const unsigned short [SETWIDTH](#)
- const unsigned short [TAGWIDTH](#)
- const uint32\_t [OFF\\_BITMASK](#)
- const uint32\_t [TAG\\_BITMASK](#)
- const uint32\_t [SET\\_BITMASK](#)
- std::unordered\_map< int, char \* > \* [mainMem](#)
- std::list< [Slot](#) > \* [sets](#)

### 6.1.1 Constructor & Destructor Documentation

**6.1.1.1 [Cache::Cache](#) ( const char \* *f*, const unsigned short *cs*, const unsigned short *a*, const unsigned short *bs* )**  
**[inline]**

[Cache](#) constructor. Instantiates new cache based on cache size, associativity, and block size. Calls init on self after instantiating member variables.

**6.1.1.2 [Cache::~~Cache](#) ( )** **[inline]**

[Cache](#) destructor

### 6.1.2 Member Function Documentation

**6.1.2.1 void [Cache::exec](#) ( )**

After tracefile has been loaded with [loadFile\(\)](#) call, [exec\(\)](#) loops through the tracefile, decodes the instructions (strings to parameters), and calls the appropriate methods based on the type of instruction.

Currently supports only store, load, and comments.

Comments (lines beginning with "//", excluding leading whitespace) are printed to STDOUT

Returns

void

**6.1.2.2 std::list< [Cache::Slot](#) >::iterator [Cache::findMatch](#) ( std::list< [Slot](#) > & *s*, const uint32\_t *address*, [CacheResult](#) & *cr*, const unsigned short *accessSize* )** **[private]**

Either finds a matching tag within a set, or will return the last item in the set. Calls popSlot to remove the relevant slot.

Side effects include setting the [CacheResult](#) hit and value appropriately.

Returns

std::list<[Slot](#)>::iterator

6.1.2.3 `const unsigned short Cache::getAssociativity ( ) const`

6.1.2.4 `const unsigned short Cache::getBlockSize ( ) const`

6.1.2.5 `const unsigned short Cache::getCacheSize ( ) const`

6.1.2.6 `const unsigned short Cache::getNumBlocks ( ) const`

6.1.2.7 `const unsigned short Cache::getNumSets ( ) const`

6.1.2.8 `void Cache::init ( ) [private]`

Initializes the sets to contain n slots, where the cache is an n-way set-associative cache

Also initializes cache memory to 0's

Returns

void

6.1.2.9 `Cache::CacheResult Cache::load ( unsigned int address, unsigned short accessSize ) [private]`

Loads data from cache, or attempts to fetch from main memory if cache miss.

Returns

[CacheResult](#)

6.1.2.10 `void Cache::loadFile ( )`

Calls [loadFile\(const char\\*\)](#) with instantiated filename

Otherwise equivalent to `loadFile(_fileName)`

Returns

void

6.1.2.11 `const bool Cache::loadFile ( const char * f )`

Opens a filestream as a member variable `_fs`

Returns

bool true if file is found

6.1.2.12 `void Cache::popSlot ( std::list< Slot > & s, std::list< Slot >::iterator & it ) [private]`

Removes a slot from a given set. Also checks if the slot was marked as dirty. If dirty, then will write back to mainMemory.

Side effect is that `s` will have one less item and it will be invalidated

Returns

void

**6.1.2.13** `Cache::CacheResult Cache::store ( unsigned int address, unsigned short accessSize, char * value )`  
`[private]`

Stores a value in cache memory in the appropriate set and block.

Returns

[CacheResult](#) which contains a bool for hit and a value (0 if miss, cached value if hit)

### 6.1.3 Member Data Documentation

**6.1.3.1** `const unsigned short Cache::_associativity` `[private]`

**6.1.3.2** `const unsigned short Cache::_blockSize` `[private]`

**6.1.3.3** `const unsigned short Cache::_cacheSize` `[private]`

**6.1.3.4** `const char* Cache::_filename` `[private]`

**6.1.3.5** `std::ifstream Cache::_fs` `[private]`

**6.1.3.6** `const unsigned short Cache::_numBlocks` `[private]`

**6.1.3.7** `const unsigned short Cache::_numSets` `[private]`

**6.1.3.8** `std::unordered_map<int,char *>* Cache::mainMem` `[private]`

**6.1.3.9** `const uint32_t Cache::OFF_BITMASK` `[private]`

**6.1.3.10** `const unsigned short Cache::OFFWIDTH` `[private]`

**6.1.3.11** `const uint32_t Cache::SET_BITMASK` `[private]`

**6.1.3.12** `std::list<Slot>* Cache::sets` `[private]`

**6.1.3.13** `const unsigned short Cache::SETWIDTH` `[private]`

**6.1.3.14** `const uint32_t Cache::TAG_BITMASK` `[private]`

**6.1.3.15** `const unsigned short Cache::TAGWIDTH` `[private]`

The documentation for this class was generated from the following files:

- [cache.h](#)
- [cache.cc](#)

## 6.2 Cache::CacheResult Struct Reference

### Public Attributes

- bool [hit](#)
- char \* [value](#)

### 6.2.1 Member Data Documentation

6.2.1.1 `bool Cache::CacheResult::hit`

6.2.1.2 `char* Cache::CacheResult::value`

The documentation for this struct was generated from the following file:

- [cache.h](#)

## 6.3 Cache::Slot Struct Reference

### Public Attributes

- `bool V`
- `bool d`
- `uint32_t fields`
- `char * data`

### 6.3.1 Member Data Documentation

6.3.1.1 `bool Cache::Slot::d`

6.3.1.2 `char* Cache::Slot::data`

6.3.1.3 `uint32_t Cache::Slot::fields`

6.3.1.4 `bool Cache::Slot::V`

The documentation for this struct was generated from the following file:

- [cache.h](#)



## Chapter 7

# File Documentation

### 7.1 cache.cc File Reference

```
#include <iostream>
#include <fstream>
#include <ostream>
#include <sstream>
#include <vector>
#include <iterator>
#include <algorithm>
#include <list>
#include <stdint>
#include <unordered_map>
#include "util.h"
#include "cache.h"
```

#### Functions

- `template<typename T >`  
  T [FromString](#) (const char \*str)

#### 7.1.1 Function Documentation

7.1.1.1 `template<typename T > T FromString ( const char * str )`

### 7.2 cache.h File Reference

```
#include <fstream>
#include <ostream>
#include <stdint>
#include <unordered_map>
#include <list>
#include <cmath>
```

#### Classes

- class [Cache](#)

- struct [Cache::Slot](#)
- struct [Cache::CacheResult](#)

## Defines

- `#define` [BUSWIDTH](#) 32

## Functions

- `template<typename T >`  
`T FromString (const char *str)`

### 7.2.1 Define Documentation

#### 7.2.1.1 `#define BUSWIDTH 32`

### 7.2.2 Function Documentation

#### 7.2.2.1 `template<typename T > T FromString ( const char * str )`

## 7.3 cachesim.cc File Reference

```
#include <iostream>
#include <sstream>
#include "cache.h"
#include "util.h"
#include "cachesim.h"
```

## Functions

- `int` [main](#) (int argc, const char \*argv[])
- `template<typename T >`  
`T FromString (const char *str)`

### 7.3.1 Function Documentation

#### 7.3.1.1 `template<typename T > T FromString ( const char * str )`

#### 7.3.1.2 `int main ( int argc, const char * argv[ ] )`

## 7.4 cachesim.h File Reference

## Functions

- `int` [main](#) (int argc, const char \*argv[])
- `template<typename T >`  
`T FromString (const char *str)`



### 7.4.1 Function Documentation

7.4.1.1 `template<typename T > T FromString ( const char * str )`

7.4.1.2 `int main ( int argc, const char * argv[] )`

## 7.5 README.md File Reference

## 7.6 util.cc File Reference

```
#include <sstream>
#include <vector>
#include <iterator>
#include <algorithm>
#include <iomanip>
#include <functional>
#include <locale>
#include "cache.h"
#include "util.h"
```

## 7.7 util.h File Reference

```
#include <vector>
#include <ostream>
#include "cache.h"
```

### Namespaces

- namespace [util](#)

### Functions

- `std::vector< std::string > util::splitLine (const std::string str, const char delim)`
- `std::ostream & util::operator<< (std::ostream &out, const Cache &c)`
- `void util::padHex (std::ostream &out, char *value, const int bytes)`
- `std::string & util::trim (std::string &s)`
- `std::string & util::ltrim (std::string &s)`
- `std::string & util::rtrim (std::string &s)`

# Index

- ~Cache
  - Cache, [12](#)
- \_associativity
  - Cache, [14](#)
- \_blockSize
  - Cache, [14](#)
- \_cacheSize
  - Cache, [14](#)
- \_filename
  - Cache, [14](#)
- \_fs
  - Cache, [14](#)
- \_numBlocks
  - Cache, [14](#)
- \_numSets
  - Cache, [14](#)
- BUSWIDTH
  - cache.h, [18](#)
- Cache, [11](#)
  - ~Cache, [12](#)
  - \_associativity, [14](#)
  - \_blockSize, [14](#)
  - \_cacheSize, [14](#)
  - \_filename, [14](#)
  - \_fs, [14](#)
  - \_numBlocks, [14](#)
  - \_numSets, [14](#)
  - Cache, [12](#)
  - exec, [12](#)
  - findMatch, [12](#)
  - getAssociativity, [12](#)
  - getBlockSize, [13](#)
  - getCacheSize, [13](#)
  - getNumBlocks, [13](#)
  - getNumSets, [13](#)
  - init, [13](#)
  - load, [13](#)
  - loadFile, [13](#)
  - mainMem, [14](#)
  - OFF\_BITMASK, [14](#)
  - OFFWIDTH, [14](#)
  - popSlot, [13](#)
  - SET\_BITMASK, [14](#)
  - SETWIDTH, [14](#)
  - sets, [14](#)
  - store, [13](#)
  - TAG\_BITMASK, [14](#)
  - TAGWIDTH, [14](#)
- cache.cc, [17](#)
  - FromString, [17](#)
- cache.h, [17](#)
  - BUSWIDTH, [18](#)
  - FromString, [18](#)
- Cache::CacheResult, [14](#)
  - hit, [15](#)
  - value, [15](#)
- Cache::Slot, [15](#)
  - d, [15](#)
  - data, [15](#)
  - fields, [15](#)
  - V, [15](#)
- cachesim.cc, [18](#)
  - FromString, [18](#)
  - main, [18](#)
- cachesim.h, [18](#)
  - FromString, [19](#)
  - main, [19](#)
- d
  - Cache::Slot, [15](#)
- data
  - Cache::Slot, [15](#)
- exec
  - Cache, [12](#)
- fields
  - Cache::Slot, [15](#)
- findMatch
  - Cache, [12](#)
- FromString
  - cache.cc, [17](#)
  - cache.h, [18](#)
  - cachesim.cc, [18](#)
  - cachesim.h, [19](#)
- getAssociativity
  - Cache, [12](#)
- getBlockSize
  - Cache, [13](#)
- getCacheSize
  - Cache, [13](#)
- getNumBlocks
  - Cache, [13](#)
- getNumSets
  - Cache, [13](#)
- hit
  - Cache::CacheResult, [15](#)

- init
  - Cache, [13](#)
- load
  - Cache, [13](#)
- loadFile
  - Cache, [13](#)
- ltrim
  - util, [9](#)
- main
  - cachesim.cc, [18](#)
  - cachesim.h, [19](#)
- mainMem
  - Cache, [14](#)
- OFF\_BITMASK
  - Cache, [14](#)
- OFFWIDTH
  - Cache, [14](#)
- operator<<
  - util, [9](#)
- padHex
  - util, [9](#)
- popSlot
  - Cache, [13](#)
- README.md, [19](#)
- rtrim
  - util, [9](#)
- SET\_BITMASK
  - Cache, [14](#)
- SETWIDTH
  - Cache, [14](#)
- sets
  - Cache, [14](#)
- splitLine
  - util, [9](#)
- store
  - Cache, [13](#)
- TAG\_BITMASK
  - Cache, [14](#)
- TAGWIDTH
  - Cache, [14](#)
- trim
  - util, [9](#)
- util, [9](#)
  - ltrim, [9](#)
  - operator<<, [9](#)
  - padHex, [9](#)
  - rtrim, [9](#)
  - splitLine, [9](#)
  - trim, [9](#)
- util.cc, [19](#)
- util.h, [19](#)
- V
  - Cache::Slot, [15](#)
- value
  - Cache::CacheResult, [15](#)