Anthony Redamonti

Jaime L Ponicki

Ang Li

Syracuse university

Homework 4

CIS-655 Advanced computer architecture

prof. mohammed abdallah

11/29/2021

1. Installation

First, navigate to the simple scalar website using a web browser. The url is: <http://simplescalar.com>. Download the following files: simplesim-3v0e.tgz, simpletools-2v0.tgz, and simpleutils-2v0.tgz.

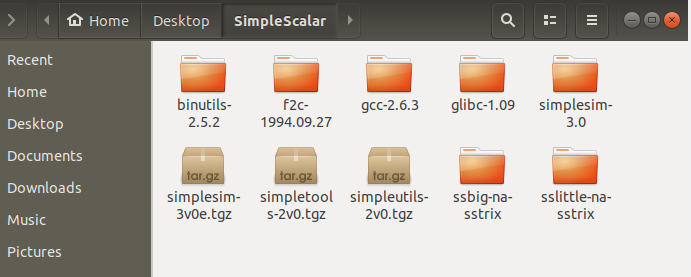
Create a folder in the Desktop named “SimpleScalar” and copy the simplesim-3v0e.tgz, simpletools-2v0.tgz, and simpleutils-2v0.tgz into the folder. Unpack each of them with the following commands:

tar -xvf simplesim-3v0e.tgz

tar -xvf simpletools-2v0.tgz

tar -xvf simpleutils-2v0.tgz

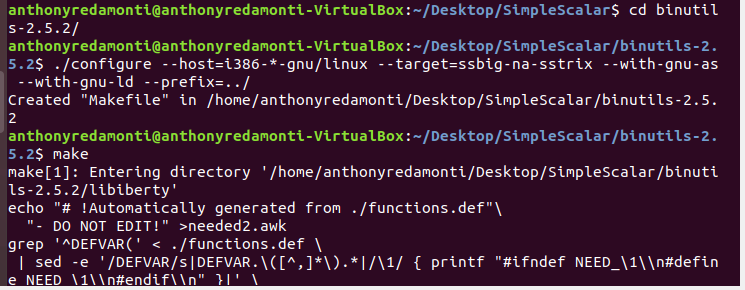
The SimpleScalar folder should look like the below image.



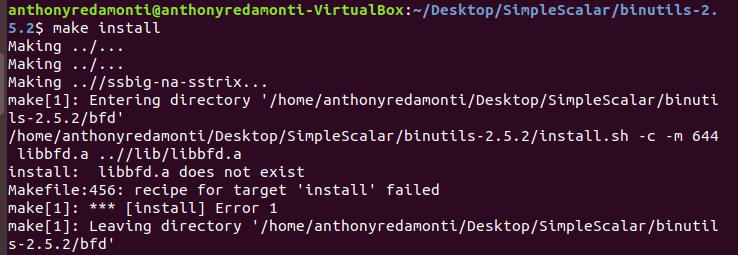
Next, navigate to the binutils-2.5.2 folder and run the configure executable to configure the custom compiler “ssbig-na-sstrix.”

cd binutils-2.5.2

./configure --host=i386-\*-gnu/linux --target=ssbig-na-sstrix --with-gnu-as --with-gnu-ld --prefix=../



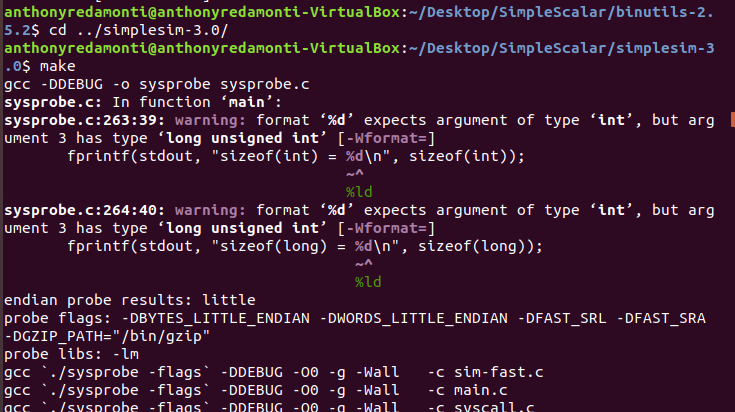
Next, make the install file using “make install” shown below.



Navigate to the parent simplesim-3.0 folder and run the make file using the following commands:

cd ../simplesim-3.0/

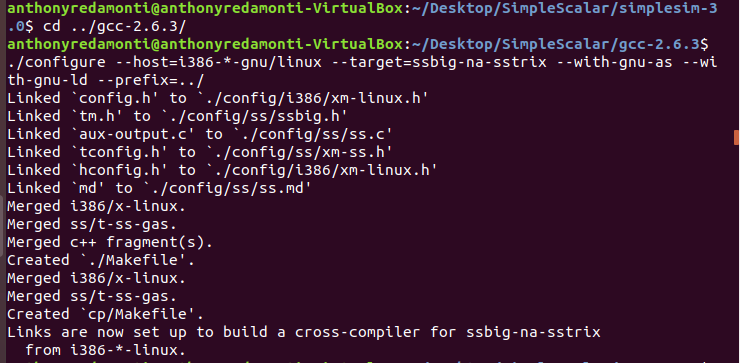
make



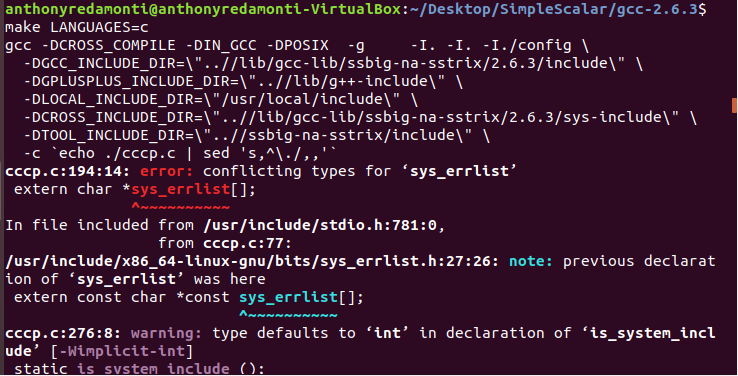
Next, run the configure executable inside of the gcc-2.6.3 folder by running the following commands:

cd ../gcc-2.6.3/

./configure --host=i386-\*-gnu/linux --target=ssbig-na-sstrix --with-gnu-as --with-gnu-ld --prefix=../



Next, run the make file using the C languages setting: “make LANGUAGES=c.”



The installation is complete.

1. Creating and testing a 4-level cache system

The team tested a separate level 4 I&D cache design against a unified level 4 I&D cache design. Levels 1 through 3 had separate I&D in both configurations.

The team found an implementation of a 3-level cache system here: <https://github.com/KaiboLiu/sim-cache-l3>. We decided to implement a 4-level cache system. To create a 4-level cache, changes were made to the sim-cache.c file. The edited file is in the appendix.

The following configuration files were created to test the two systems: cache\_4a.cfg and cache\_4b.cfg.

cache\_4a.cfg:

# l1 inst cache config, i.e., {<config>|dl1|dl2|none}

-cache:il1 il1:64:64:1:l

# l2 instruction cache config, i.e., {<config>|dl2|none}

-cache:il2 il2:128:64:1:l;

# l3 instruction cache config, i.e., {<config>|dl3|none}

-cache:il3 il3:256:64:1:l;

# l4 instruction cache config, i.e., {<config>|dl4|none}

-cache:il4 il4:512:64:1:l;

# l1 data cache config, i.e., {<config>|none}

-cache:dl1 dl1:64:64:1:l

# l2 data cache config, i.e., {<config>|none}

-cache:dl2 dl2:128:64:1:l

# l3 data cache config, i.e., {<config>|none}

-cache:dl3 dl3:256:64:1:l

# l4 data cache config, i.e., {<config>|none}

-cache:dl4 dl4:512:64:1:l

# instruction TLB config, i.e., {<config>|none}

-tlb:itlb none

# data TLB config, i.e., {<config>|none}

-tlb:dtlb none

cache\_4b.cfg:

# l1 inst cache config, i.e., {<config>|dl1|dl2|none}

-cache:il1 il1:64:64:1:l

# l2 instruction cache config, i.e., {<config>|dl2|none}

-cache:il2 il2:128:64:1:l;

# l3 instruction cache config, i.e., {<config>|dl3|none}

-cache:il3 il3:256:64:1:l;

# l4 instruction cache config, i.e., {<config>|dl4|none}

-cache:il4 dl4

# l1 data cache config, i.e., {<config>|none}

-cache:dl1 dl1:64:64:1:l

# l2 data cache config, i.e., {<config>|none}

-cache:dl2 dl2:128:64:1:l

# l3 data cache config, i.e., {<config>|none}

-cache:dl3 dl3:256:64:1:l

# l4 data cache config, i.e., {<config>|none}

-cache:dl4 dl4:1024:64:1:l

# instruction TLB config, i.e., {<config>|none}

-tlb:itlb none

# data TLB config, i.e., {<config>|none}

-tlb:dtlb none

The outputs cache\_4a.out and cache\_4b.out were created using the following commands:





The contents of both output files are in the appendix. The salient differences in the files are displayed in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Cache Level 4** | | | |
| Separate I&D | | | Unified I&D |
|  | Instruction | Data |
| Accesses | 10531 | 399 | 10930 |
| Hits | 6141 | 101 | 8982 |
| Misses | 4390 | 298 | 1948 |
| Replacements | 3893 | 12 | 1169 |
| Writebacks | 0 | 9 | 55 |
| Invalidations | 0 | 0 | 0 |
| Miss Rate | 0.4169 | 0.7469 | 0.1782 |
| Replacement Rate | 0.3697 | 0.0301 | 0.1070 |
| Writeback Rate | 0.0000 | 0.0226 | 0.0050 |
| Invalidation Rate | 0.0000 | 0.0000 | 0.0000 |

The total number of memory accesses for level 4 are the same for both configurations: data = 399, instruction = 10531 for separate I&D and 10930 for unified I&D. The miss rate and replacement rate for separate is much higher than unified.

1. Conclusion

In the chart above, we have compared the separate and unified Instruction and Data statistics. In general, separate Instructions and Data is more advantageous in terms of pipelining. It provides more efficient access to the caches and can potentially impact execution time and throughput. The split design allows for the instruction cache to be placed close to the instruction fetch unit and the data cache close to the memory unit, simultaneously reducing the latencies of both.

As the collected data shows, the hit rate is higher with a unified Instruction and Data structure, which matches the commonly accepted performance characteristics in literature. Unified caches are typically used in systems with limited space and expense is available. This is achieved as unified caches balance the load between instructions and data automatically.

In terms of structural design choices made in industry based on multi-level caches, typical processor architectures have three levels of cache. Faster RAM speeds seem to have made a fourth level cache obsolete. A balance is required to supply a meaningful amount of bandwidth between the core and caches. If caches become too big, speed can slow. Thus, the balance of three level caches was reached.

Appendix:

sim-cache.c

/\* sim-cache.c - sample cache simulator implementation \*/

/\* SimpleScalar(TM) Tool Suite

\* Copyright (C) 1994-2003 by Todd M. Austin, Ph.D. and SimpleScalar, LLC.

\* All Rights Reserved.

\*

\* THIS IS A LEGAL DOCUMENT, BY USING SIMPLESCALAR,

\* YOU ARE AGREEING TO THESE TERMS AND CONDITIONS.

\*

\* No portion of this work may be used by any commercial entity, or for any

\* commercial purpose, without the prior, written permission of SimpleScalar,

\* LLC (info@simplescalar.com). Nonprofit and noncommercial use is permitted

\* as described below.

\*

\* 1. SimpleScalar is provided AS IS, with no warranty of any kind, express

\* or implied. The user of the program accepts full responsibility for the

\* application of the program and the use of any results.

\*

\* 2. Nonprofit and noncommercial use is encouraged. SimpleScalar may be

\* downloaded, compiled, executed, copied, and modified solely for nonprofit,

\* educational, noncommercial research, and noncommercial scholarship

\* purposes provided that this notice in its entirety accompanies all copies.

\* Copies of the modified software can be delivered to persons who use it

\* solely for nonprofit, educational, noncommercial research, and

\* noncommercial scholarship purposes provided that this notice in its

\* entirety accompanies all copies.

\*

\* 3. ALL COMMERCIAL USE, AND ALL USE BY FOR PROFIT ENTITIES, IS EXPRESSLY

\* PROHIBITED WITHOUT A LICENSE FROM SIMPLESCALAR, LLC (info@simplescalar.com).

\*

\* 4. No nonprofit user may place any restrictions on the use of this software,

\* including as modified by the user, by any other authorized user.

\*

\* 5. Noncommercial and nonprofit users may distribute copies of SimpleScalar

\* in compiled or executable form as set forth in Section 2, provided that

\* either: (A) it is accompanied by the corresponding machine-readable source

\* code, or (B) it is accompanied by a written offer, with no time limit, to

\* give anyone a machine-readable copy of the corresponding source code in

\* return for reimbursement of the cost of distribution. This written offer

\* must permit verbatim duplication by anyone, or (C) it is distributed by

\* someone who received only the executable form, and is accompanied by a

\* copy of the written offer of source code.

\*

\* 6. SimpleScalar was developed by Todd M. Austin, Ph.D. The tool suite is

\* currently maintained by SimpleScalar LLC (info@simplescalar.com). US Mail:

\* 2395 Timbercrest Court, Ann Arbor, MI 48105.

\*

\* Copyright (C) 1994-2003 by Todd M. Austin, Ph.D. and SimpleScalar, LLC.

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <assert.h>

#include "host.h"

#include "misc.h"

#include "machine.h"

#include "regs.h"

#include "memory.h"

#include "cache.h"

#include "loader.h"

#include "syscall.h"

#include "dlite.h"

#include "sim.h"

/\*

\* This file implements a functional cache simulator. Cache statistics are

\* generated for a user-selected cache and TLB configuration, which may include

\* up to two levels of instruction and data cache (with any levels unified),

\* and one level of instruction and data TLBs. No timing information is

\* generated (hence the distinction, "functional" simulator).

\*/

/\* simulated registers \*/

static struct regs\_t regs;

/\* simulated memory \*/

static struct mem\_t \*mem = NULL;

/\* track number of insn and refs \*/

static counter\_t sim\_num\_refs = 0;

/\* maximum number of inst's to execute \*/

static unsigned int max\_insts;

/\* level 1 instruction cache, entry level instruction cache \*/

static struct cache\_t \*cache\_il1 = NULL;

/\* level 2 instruction cache \*/

static struct cache\_t \*cache\_il2 = NULL;

/\* level 3 instruction cache \*/

static struct cache\_t \*cache\_il3 = NULL;

/\* level 4 instruction cache \*/

static struct cache\_t \*cache\_il4 = NULL;

/\* level 1 data cache, entry level data cache \*/

static struct cache\_t \*cache\_dl1 = NULL;

/\* level 2 data cache \*/

static struct cache\_t \*cache\_dl2 = NULL;

/\* level 3 data cache \*/

static struct cache\_t \*cache\_dl3 = NULL;

/\* level 4 data cache \*/

static struct cache\_t \*cache\_dl4 = NULL;

/\* instruction TLB \*/

static struct cache\_t \*itlb = NULL;

/\* data TLB \*/

static struct cache\_t \*dtlb = NULL;

/\* text-based stat profiles \*/

#define MAX\_PCSTAT\_VARS 8

static struct stat\_stat\_t \*pcstat\_stats[MAX\_PCSTAT\_VARS];

static counter\_t pcstat\_lastvals[MAX\_PCSTAT\_VARS];

static struct stat\_stat\_t \*pcstat\_sdists[MAX\_PCSTAT\_VARS];

/\* wedge all stat values into a counter\_t \*/

#define STATVAL(STAT) \

((STAT)->sc == sc\_int \

? (counter\_t)\*((STAT)->variant.for\_int.var) \

: ((STAT)->sc == sc\_uint \

? (counter\_t)\*((STAT)->variant.for\_uint.var) \

: ((STAT)->sc == sc\_counter \

? \*((STAT)->variant.for\_counter.var) \

: (panic("bad stat class"), 0))))

/\* l1 data cache l1 block miss handler function \*/

static unsigned int /\* latency of block access \*/

dl1\_access\_fn(enum mem\_cmd cmd, /\* access cmd, Read or Write \*/

md\_addr\_t baddr, /\* block address to access \*/

int bsize, /\* size of block to access \*/

struct cache\_blk\_t \*blk, /\* ptr to block in upper level \*/

tick\_t now) /\* time of access \*/

{

if (cache\_dl2)

{

/\* access next level of data cache hierarchy \*/

return cache\_access(cache\_dl2, cmd, baddr, NULL, bsize,

/\* now \*/now, /\* pudata \*/NULL, /\* repl addr \*/NULL);

}

else

{

/\* access main memory, which is always done in the main simulator loop \*/

return /\* access latency, ignored \*/1;

}

}

/\* l2 data cache block miss handler function \*/

static unsigned int /\* latency of block access \*/

dl2\_access\_fn(enum mem\_cmd cmd, /\* access cmd, Read or Write \*/

md\_addr\_t baddr, /\* block address to access \*/

int bsize, /\* size of block to access \*/

struct cache\_blk\_t \*blk, /\* ptr to block in upper level \*/

tick\_t now) /\* time of access \*/

{

if (cache\_dl3)

{

/\* access next level of data cache hierarchy \*/

return cache\_access(cache\_dl3, cmd, baddr, NULL, bsize,

/\* now \*/now, /\* pudata \*/NULL, /\* repl addr \*/NULL);

}

else

{

/\* access main memory, which is always done in the main simulator loop \*/

return /\* access latency, ignored \*/1;

}

}

/\* l3 data cache block miss handler function \*/

static unsigned int /\* latency of block access \*/

dl3\_access\_fn(enum mem\_cmd cmd, /\* access cmd, Read or Write \*/

md\_addr\_t baddr, /\* block address to access \*/

int bsize, /\* size of block to access \*/

struct cache\_blk\_t \*blk, /\* ptr to block in upper level \*/

tick\_t now) /\* time of access \*/

{

if (cache\_dl4)

{

/\* access next level of data cache hierarchy \*/

return cache\_access(cache\_dl4, cmd, baddr, NULL, bsize,

/\* now \*/now, /\* pudata \*/NULL, /\* repl addr \*/NULL);

}

else

{

/\* access main memory, which is always done in the main simulator loop \*/

return /\* access latency, ignored \*/1;

}

}

/\* l4 data cache block miss handler function \*/

static unsigned int /\* latency of block access \*/

dl4\_access\_fn(enum mem\_cmd cmd, /\* access cmd, Read or Write \*/

md\_addr\_t baddr, /\* block address to access \*/

int bsize, /\* size of block to access \*/

struct cache\_blk\_t \*blk, /\* ptr to block in upper level \*/

tick\_t now) /\* time of access \*/

{

/\* this is a miss to the lowest level, so access main memory, which is

always done in the main simulator loop \*/

return /\* access latency, ignored \*/1;

}

/\* l1 inst cache l1 block miss handler function \*/

static unsigned int /\* latency of block access \*/

il1\_access\_fn(enum mem\_cmd cmd, /\* access cmd, Read or Write \*/

md\_addr\_t baddr, /\* block address to access \*/

int bsize, /\* size of block to access \*/

struct cache\_blk\_t \*blk, /\* ptr to block in upper level \*/

tick\_t now) /\* time of access \*/

{

if (cache\_il2)

{

/\* access next level of inst cache hierarchy \*/

return cache\_access(cache\_il2, cmd, baddr, NULL, bsize,

/\* now \*/now, /\* pudata \*/NULL, /\* repl addr \*/NULL);

}

else

{

/\* access main memory, which is always done in the main simulator loop \*/

return /\* access latency, ignored \*/1;

}

}

/\* l3 inst cache block miss handler function \*/

static unsigned int /\* latency of block access \*/

il3\_access\_fn(enum mem\_cmd cmd, /\* access cmd, Read or Write \*/

md\_addr\_t baddr, /\* block address to access \*/

int bsize, /\* size of block to access \*/

struct cache\_blk\_t \*blk, /\* ptr to block in upper level \*/

tick\_t now) /\* time of access \*/

{

if (cache\_il4)

{

/\* access next level of inst cache hierarchy \*/

return cache\_access(cache\_il4, cmd, baddr, NULL, bsize,

/\* now \*/now, /\* pudata \*/NULL, /\* repl addr \*/NULL);

}

else

{

/\* access main memory, which is always done in the main simulator loop \*/

return /\* access latency, ignored \*/1;

}

}

/\* l4 inst cache block miss handler function \*/

static unsigned int /\* latency of block access \*/

il4\_access\_fn(enum mem\_cmd cmd, /\* access cmd, Read or Write \*/

md\_addr\_t baddr, /\* block address to access \*/

int bsize, /\* size of block to access \*/

struct cache\_blk\_t \*blk, /\* ptr to block in upper level \*/

tick\_t now) /\* time of access \*/

{

/\* this is a miss to the lowest level, so access main memory, which is

always done in the main simulator loop \*/

return /\* access latency, ignored \*/1;

}

/\* inst cache block miss handler function \*/

static unsigned int /\* latency of block access \*/

itlb\_access\_fn(enum mem\_cmd cmd, /\* access cmd, Read or Write \*/

md\_addr\_t baddr, /\* block address to access \*/

int bsize, /\* size of block to access \*/

struct cache\_blk\_t \*blk, /\* ptr to block in upper level \*/

tick\_t now) /\* time of access \*/

{

md\_addr\_t \*phy\_page\_ptr = (md\_addr\_t \*)blk->user\_data;

/\* no real memory access, however, should have user data space attached \*/

assert(phy\_page\_ptr);

/\* fake translation, for now... \*/

\*phy\_page\_ptr = 0;

return /\* access latency, ignored \*/1;

}

/\* data cache block miss handler function \*/

static unsigned int /\* latency of block access \*/

dtlb\_access\_fn(enum mem\_cmd cmd, /\* access cmd, Read or Write \*/

md\_addr\_t baddr, /\* block address to access \*/

int bsize, /\* size of block to access \*/

struct cache\_blk\_t \*blk, /\* ptr to block in upper level \*/

tick\_t now) /\* time of access \*/

{

md\_addr\_t \*phy\_page\_ptr = (md\_addr\_t \*)blk->user\_data;

/\* no real memory access, however, should have user data space attached \*/

assert(phy\_page\_ptr);

/\* fake translation, for now... \*/

\*phy\_page\_ptr = 0;

return /\* access latency, ignored \*/1;

}

/\* cache/TLB options \*/

static char \*cache\_dl1\_opt /\* = "none" \*/;

static char \*cache\_dl2\_opt /\* = "none" \*/;

static char \*cache\_dl3\_opt /\* = "none" \*/;

static char \*cache\_dl4\_opt /\* = "none" \*/;

static char \*cache\_il1\_opt /\* = "none" \*/;

static char \*cache\_il2\_opt /\* = "none" \*/;

static char \*cache\_il3\_opt /\* = "none" \*/;

static char \*cache\_il4\_opt /\* = "none" \*/;

static char \*itlb\_opt /\* = "none" \*/;

static char \*dtlb\_opt /\* = "none" \*/;

static int flush\_on\_syscalls /\* = FALSE \*/;

static int compress\_icache\_addrs /\* = FALSE \*/;

/\* text-based stat profiles \*/

static int pcstat\_nelt = 0;

static char \*pcstat\_vars[MAX\_PCSTAT\_VARS];

/\* convert 64-bit inst text addresses to 32-bit inst equivalents \*/

#ifdef TARGET\_PISA

#define IACOMPRESS(A) \

(compress\_icache\_addrs ? ((((A) - ld\_text\_base) >> 1) + ld\_text\_base) : (A))

#define ISCOMPRESS(SZ) \

(compress\_icache\_addrs ? ((SZ) >> 1) : (SZ))

#else /\* !TARGET\_PISA \*/

#define IACOMPRESS(A) (A)

#define ISCOMPRESS(SZ) (SZ)

#endif /\* TARGET\_PISA \*/

/\* Register simulator-specific options \*/

void

sim\_reg\_options(struct opt\_odb\_t \*odb) /\* options database \*/

{

opt\_reg\_header(odb,

"sim-cache: This simulator implements a functional cache simulator. Cache\n"

"statistics are generated for a user-selected cache and TLB configuration,\n"

"which may include up to four levels of instruction and data cache (with any\n"

"levels unified), and one level of instruction and data TLBs. No timing\n"

"information is generated.\n"

);

/\* instruction limit \*/

opt\_reg\_uint(odb, "-max:inst", "maximum number of inst's to execute",

&max\_insts, /\* default \*/0,

/\* print \*/TRUE, /\* format \*/NULL);

opt\_reg\_string(odb, "-cache:dl1",

"l1 data cache config, i.e., {<config>|none}",

&cache\_dl1\_opt, "dl1:256:32:1:l", /\* print \*/TRUE, NULL);

opt\_reg\_note(odb,

" The cache config parameter <config> has the following format:\n"

"\n"

" <name>:<nsets>:<bsize>:<assoc>:<repl>\n"

"\n"

" <name> - name of the cache being defined\n"

" <nsets> - number of sets in the cache\n"

" <bsize> - block size of the cache\n"

" <assoc> - associativity of the cache\n"

" <repl> - block replacement strategy, 'l'-LRU, 'f'-FIFO, 'r'-random\n"

"\n"

" Examples: -cache:dl1 dl1:4096:32:1:l\n"

" -dtlb dtlb:128:4096:32:r\n"

);

opt\_reg\_string(odb, "-cache:dl2",

"l2 data cache config, i.e., {<config>|none}",

&cache\_dl2\_opt, "ul2:256:64:4:l", /\* print \*/TRUE, NULL);

opt\_reg\_string(odb, "-cache:dl3",

"l3 data cache config, i.e., {<config>|none}",

&cache\_dl3\_opt, "ul3:512:64:4:l", /\* print \*/TRUE, NULL);

opt\_reg\_string(odb, "-cache:dl4",

"l4 data cache config, i.e., {<config>|none}",

&cache\_dl4\_opt, "ul4:1024:64:4:l", /\* print \*/TRUE, NULL);

opt\_reg\_string(odb, "-cache:il1",

"l1 inst cache config, i.e., {<config>|dl1|dl2|dl3|dl4|none}",

&cache\_il1\_opt, "il1:256:32:1:l", /\* print \*/TRUE, NULL);

opt\_reg\_note(odb,

" Cache levels can be unified by pointing a level of the instruction cache\n"

" hierarchy at the data cache hiearchy using the \"dl1\" and \"dl2\" cache\n"

" configuration arguments. Most sensible combinations are supported, e.g.,\n"

"\n"

" A unified l2 cache (il2 is pointed at dl2):\n"

" -cache:il1 il1:128:64:1:l -cache:il2 dl2\n"

" -cache:dl1 dl1:256:32:1:l -cache:dl2 ul2:1024:64:2:l\n"

"\n"

" Or, a fully unified cache hierarchy (il1 pointed at dl1):\n"

" -cache:il1 dl1\n"

" -cache:dl1 ul1:256:32:1:l -cache:dl2 ul2:1024:64:2:l\n"

);

opt\_reg\_string(odb, "-cache:il2",

"l2 instruction cache config, i.e., {<config>|dl2|dl3|dl4|none}",

&cache\_il2\_opt, "dl2", /\* print \*/TRUE, NULL);

opt\_reg\_string(odb, "-cache:il3",

"l3 instruction cache config, i.e., {<config>|dl3|dl4|none}",

&cache\_il3\_opt, "dl3", /\* print \*/TRUE, NULL);

opt\_reg\_string(odb, "-cache:il4",

"l4 instruction cache config, i.e., {<config>|dl4|none}",

&cache\_il4\_opt, "dl4", /\* print \*/TRUE, NULL);

opt\_reg\_string(odb, "-tlb:itlb",

"instruction TLB config, i.e., {<config>|none}",

&itlb\_opt, "itlb:16:4096:4:l", /\* print \*/TRUE, NULL);

opt\_reg\_string(odb, "-tlb:dtlb",

"data TLB config, i.e., {<config>|none}",

&dtlb\_opt, "dtlb:32:4096:4:l", /\* print \*/TRUE, NULL);

opt\_reg\_flag(odb, "-flush", "flush caches on system calls",

&flush\_on\_syscalls, /\* default \*/FALSE, /\* print \*/TRUE, NULL);

opt\_reg\_flag(odb, "-cache:icompress",

"convert 64-bit inst addresses to 32-bit inst equivalents",

&compress\_icache\_addrs, /\* default \*/FALSE,

/\* print \*/TRUE, NULL);

opt\_reg\_string\_list(odb, "-pcstat",

"profile stat(s) against text addr's (mult uses ok)",

pcstat\_vars, MAX\_PCSTAT\_VARS, &pcstat\_nelt, NULL,

/\* !print \*/FALSE, /\* format \*/NULL, /\* accrue \*/TRUE);

}

/\* check simulator-specific option values \*/

void

sim\_check\_options(struct opt\_odb\_t \*odb, /\* options database \*/

int argc, char \*\*argv) /\* command line arguments \*/

{

char name[128], c;

int nsets, bsize, assoc;

/\* use a level 1 D-cache? \*/

if (!mystricmp(cache\_dl1\_opt, "none"))

{

cache\_dl1 = NULL;

/\* the level 2 D-cache cannot be defined \*/

if (strcmp(cache\_dl2\_opt, "none"))

fatal("the l1 data cache must defined if the l2 cache is defined");

cache\_dl2 = NULL;

/\* the level 3 D-cache cannot be defined \*/

if (strcmp(cache\_dl3\_opt, "none"))

fatal("the l1 data cache must defined if the l3 cache is defined");

cache\_dl3 = NULL;

/\* the level 4 D-cache cannot be defined \*/

if (strcmp(cache\_dl4\_opt, "none"))

fatal("the l1 data cache must defined if the l4 cache is defined");

cache\_dl4 = NULL;

}

else /\* dl1 is defined \*/

{

if (sscanf(cache\_dl1\_opt, "%[^:]:%d:%d:%d:%c",

name, &nsets, &bsize, &assoc, &c) != 5)

fatal("bad l1 D-cache parms: <name>:<nsets>:<bsize>:<assoc>:<repl>");

cache\_dl1 = cache\_create(name, nsets, bsize, /\* balloc \*/FALSE,

/\* usize \*/0, assoc, cache\_char2policy(c),

dl1\_access\_fn, /\* hit latency \*/1);

/\* is the level 2 D-cache defined? \*/

if (!mystricmp(cache\_dl2\_opt, "none"))

{

cache\_dl2 = NULL;

/\* the level 3 D-cache cannot be defined \*/

if (strcmp(cache\_dl3\_opt, "none"))

fatal("the l2 data cache must defined if the l3 cache is defined");

cache\_dl3 = NULL;

/\* the level 4 D-cache cannot be defined \*/

if (strcmp(cache\_dl4\_opt, "none"))

fatal("the l2 data cache must defined if the l4 cache is defined");

cache\_dl4 = NULL;

}

else /\*dl2 is defined \*/

{

if (sscanf(cache\_dl2\_opt, "%[^:]:%d:%d:%d:%c",

name, &nsets, &bsize, &assoc, &c) != 5)

fatal("bad l2 D-cache parms: "

"<name>:<nsets>:<bsize>:<assoc>:<repl>");

cache\_dl2 = cache\_create(name, nsets, bsize, /\* balloc \*/FALSE,

/\* usize \*/0, assoc, cache\_char2policy(c),

dl2\_access\_fn, /\* hit latency \*/1);

/\* is the level 3 D-cache defined? \*/

if (!mystricmp(cache\_dl3\_opt, "none"))

{

cache\_dl3 = NULL;

/\* the level 4 D-cache cannot be defined \*/

if (strcmp(cache\_dl4\_opt, "none"))

fatal("the l3 data cache must defined if the l4 cache is defined");

cache\_dl4 = NULL;

}

else /\*dl3 is defined \*/

{

if (sscanf(cache\_dl3\_opt, "%[^:]:%d:%d:%d:%c",

name, &nsets, &bsize, &assoc, &c) != 5)

fatal("bad l3 D-cache parms: "

"<name>:<nsets>:<bsize>:<assoc>:<repl>");

cache\_dl3 = cache\_create(name, nsets, bsize, /\* balloc \*/FALSE,

/\* usize \*/0, assoc, cache\_char2policy(c),

dl3\_access\_fn, /\* hit latency \*/1);

/\* is the level 4 D-cache defined? \*/

if (!mystricmp(cache\_dl4\_opt, "none"))

cache\_dl4 = NULL;

else /\*dl4 is defined \*/

{

if (sscanf(cache\_dl4\_opt, "%[^:]:%d:%d:%d:%c",

name, &nsets, &bsize, &assoc, &c) != 5)

fatal("bad l4 D-cache parms: "

"<name>:<nsets>:<bsize>:<assoc>:<repl>");

cache\_dl4 = cache\_create(name, nsets, bsize, /\* balloc \*/FALSE,

/\* usize \*/0, assoc, cache\_char2policy(c),

dl4\_access\_fn, /\* hit latency \*/1);

}

}

}

}

/\* use a level 1 I-cache? \*/

if (!mystricmp(cache\_il1\_opt, "none"))

{

cache\_il1 = NULL;

/\* the level 2 I-cache cannot be defined \*/

if (strcmp(cache\_il2\_opt, "none"))

fatal("the l1 inst cache must defined if the l2 cache is defined");

cache\_il2 = NULL;

/\* the level 3 I-cache cannot be defined \*/

if (strcmp(cache\_il3\_opt, "none"))

fatal("the l1 inst cache must defined if the l3 cache is defined");

cache\_il3 = NULL;

/\* the level 4 I-cache cannot be defined \*/

if (strcmp(cache\_il4\_opt, "none"))

fatal("the l1 inst cache must defined if the l4 cache is defined");

cache\_il4 = NULL;

}

else if (!mystricmp(cache\_il1\_opt, "dl1"))

{

if (!cache\_dl1)

fatal("I-cache l1 cannot access D-cache l1 as it's undefined");

cache\_il1 = cache\_dl1;

/\* the level 2 I-cache cannot be defined \*/

if (strcmp(cache\_il2\_opt, "none"))

fatal("the l1 inst cache must defined if the l2 cache is defined");

cache\_il2 = NULL;

/\* the level 3 I-cache cannot be defined \*/

if (strcmp(cache\_il3\_opt, "none"))

fatal("the l1 inst cache must defined if the l3 cache is defined");

cache\_il3 = NULL;

/\* the level 4 I-cache cannot be defined \*/

if (strcmp(cache\_il4\_opt, "none"))

fatal("the l1 inst cache must defined if the l4 cache is defined");

cache\_il4 = NULL;

}

else if (!mystricmp(cache\_il1\_opt, "dl2"))

{

if (!cache\_dl2)

fatal("I-cache l1 cannot access D-cache l2 as it's undefined");

cache\_il1 = cache\_dl2;

/\* the level 2 I-cache cannot be defined \*/

if (strcmp(cache\_il2\_opt, "none"))

fatal("the l1 inst cache must defined if the l2 cache is defined");

cache\_il2 = NULL;

/\* the level 3 I-cache cannot be defined \*/

if (strcmp(cache\_il3\_opt, "none"))

fatal("the l1 inst cache must defined if the l3 cache is defined");

cache\_il3 = NULL;

/\* the level 4 I-cache cannot be defined \*/

if (strcmp(cache\_il4\_opt, "none"))

fatal("the l1 inst cache must defined if the l4 cache is defined");

cache\_il4 = NULL;

}

else if (!mystricmp(cache\_il1\_opt, "dl3"))

{

if (!cache\_dl3)

fatal("I-cache l1 cannot access D-cache l3 as it's undefined");

cache\_il1 = cache\_dl2;

/\* the level 2 I-cache cannot be defined \*/

if (strcmp(cache\_il2\_opt, "none"))

fatal("the l1 inst cache must defined if the l2 cache is defined");

cache\_il2 = NULL;

/\* the level 3 I-cache cannot be defined \*/

if (strcmp(cache\_il3\_opt, "none"))

fatal("the l1 inst cache must defined if the l3 cache is defined");

cache\_il3 = NULL;

/\* the level 4 I-cache cannot be defined \*/

if (strcmp(cache\_il4\_opt, "none"))

fatal("the l1 inst cache must defined if the l4 cache is defined");

cache\_il4 = NULL;

}

else if (!mystricmp(cache\_il1\_opt, "dl4"))

{

if (!cache\_dl4)

fatal("I-cache l1 cannot access D-cache l4 as it's undefined");

cache\_il1 = cache\_dl2;

/\* the level 2 I-cache cannot be defined \*/

if (strcmp(cache\_il2\_opt, "none"))

fatal("the l1 inst cache must defined if the l2 cache is defined");

cache\_il2 = NULL;

/\* the level 3 I-cache cannot be defined \*/

if (strcmp(cache\_il3\_opt, "none"))

fatal("the l1 inst cache must defined if the l3 cache is defined");

cache\_il3 = NULL;

/\* the level 4 I-cache cannot be defined \*/

if (strcmp(cache\_il4\_opt, "none"))

fatal("the l1 inst cache must defined if the l4 cache is defined");

cache\_il4 = NULL;

}

else /\* il1 is defined \*/

{

if (sscanf(cache\_il1\_opt, "%[^:]:%d:%d:%d:%c",

name, &nsets, &bsize, &assoc, &c) != 5)

fatal("bad l1 I-cache parms: <name>:<nsets>:<bsize>:<assoc>:<repl>");

cache\_il1 = cache\_create(name, nsets, bsize, /\* balloc \*/FALSE,

/\* usize \*/0, assoc, cache\_char2policy(c),

il1\_access\_fn, /\* hit latency \*/1);

/\* is the level 2 D-cache defined? \*/

if (!mystricmp(cache\_il2\_opt, "none"))

{

cache\_il2 = NULL;

/\* the level 3 I-cache cannot be defined \*/

if (strcmp(cache\_il3\_opt, "none"))

fatal("the l2 inst cache must defined if the l3 cache is defined");

cache\_il3 = NULL;

/\* the level 4 I-cache cannot be defined \*/

if (strcmp(cache\_il4\_opt, "none"))

fatal("the l2 inst cache must defined if the l4 cache is defined");

cache\_il4 = NULL;

}

else if (!mystricmp(cache\_il2\_opt, "dl2"))

{

if (!cache\_dl2)

fatal("I-cache l2 cannot access D-cache l2 as it's undefined");

cache\_il2 = cache\_dl2;

/\* the level 3 I-cache cannot be defined \*/

if (strcmp(cache\_il3\_opt, "none"))

fatal("the l2 inst cache must defined if the l3 cache is defined");

cache\_il3 = NULL;

/\* the level 4 I-cache cannot be defined \*/

if (strcmp(cache\_il4\_opt, "none"))

fatal("the l2 inst cache must defined if the l4 cache is defined");

cache\_il4 = NULL;

}

else /\* il2 is defined \*/

{

if (sscanf(cache\_il2\_opt, "%[^:]:%d:%d:%d:%c",

name, &nsets, &bsize, &assoc, &c) != 5)

fatal("bad l2 I-cache parms: <name>:<nsets>:<bsize>:<assoc>:<repl>");

cache\_il2 = cache\_create(name, nsets, bsize, /\* balloc \*/FALSE,

/\* usize \*/0, assoc, cache\_char2policy(c),

il2\_access\_fn, /\* hit latency \*/1);

/\* is the level 3 D-cache defined? \*/

if (!mystricmp(cache\_il3\_opt, "none"))

{ cache\_il3 = NULL; }

else if (!mystricmp(cache\_il3\_opt, "dl3"))

{

if (!cache\_dl3)

fatal("I-cache l3 cannot access D-cache l3 as it's undefined");

cache\_il3 = cache\_dl3;

/\* the level 4 I-cache cannot be defined \*/

if (strcmp(cache\_il4\_opt, "none"))

fatal("the l3 inst cache must defined if the l4 cache is defined");

cache\_il4 = NULL;

}

else

{

if (sscanf(cache\_il3\_opt, "%[^:]:%d:%d:%d:%c",

name, &nsets, &bsize, &assoc, &c) != 5)

fatal("bad l3 I-cache parms: "

"<name>:<nsets>:<bsize>:<assoc>:<repl>");

cache\_il3 = cache\_create(name, nsets, bsize, /\* balloc \*/FALSE,

/\* usize \*/0, assoc, cache\_char2policy(c),

il3\_access\_fn, /\* hit latency \*/1);

/\* is the level 4 D-cache defined? \*/

if (!mystricmp(cache\_il4\_opt, "none"))

{ cache\_il4 = NULL; }

else if (!mystricmp(cache\_il4\_opt, "dl4"))

{

if (!cache\_dl4)

fatal("I-cache l4 cannot access D-cache l4 as it's undefined");

cache\_il4 = cache\_dl4;

}

else

{

if (sscanf(cache\_il4\_opt, "%[^:]:%d:%d:%d:%c",

name, &nsets, &bsize, &assoc, &c) != 5)

fatal("bad l4 I-cache parms: "

"<name>:<nsets>:<bsize>:<assoc>:<repl>");

cache\_il4 = cache\_create(name, nsets, bsize, /\* balloc \*/FALSE,

/\* usize \*/0, assoc, cache\_char2policy(c),

il4\_access\_fn, /\* hit latency \*/1);

}

}

}

}

/\* use an I-TLB? \*/

if (!mystricmp(itlb\_opt, "none"))

itlb = NULL;

else

{

if (sscanf(itlb\_opt, "%[^:]:%d:%d:%d:%c",

name, &nsets, &bsize, &assoc, &c) != 5)

fatal("bad TLB parms: <name>:<nsets>:<page\_size>:<assoc>:<repl>");

itlb = cache\_create(name, nsets, bsize, /\* balloc \*/FALSE,

/\* usize \*/sizeof(md\_addr\_t), assoc,

cache\_char2policy(c), itlb\_access\_fn,

/\* hit latency \*/1);

}

/\* use a D-TLB? \*/

if (!mystricmp(dtlb\_opt, "none"))

dtlb = NULL;

else

{

if (sscanf(dtlb\_opt, "%[^:]:%d:%d:%d:%c",

name, &nsets, &bsize, &assoc, &c) != 5)

fatal("bad TLB parms: <name>:<nsets>:<page\_size>:<assoc>:<repl>");

dtlb = cache\_create(name, nsets, bsize, /\* balloc \*/FALSE,

/\* usize \*/sizeof(md\_addr\_t), assoc,

cache\_char2policy(c), dtlb\_access\_fn,

/\* hit latency \*/1);

}

}

/\* initialize the simulator \*/

void

sim\_init(void)

{

sim\_num\_refs = 0;

/\* allocate and initialize register file \*/

regs\_init(&regs);

/\* allocate and initialize memory space \*/

mem = mem\_create("mem");

mem\_init(mem);

}

/\* local machine state accessor \*/

static char \* /\* err str, NULL for no err \*/

cache\_mstate\_obj(FILE \*stream, /\* output stream \*/

char \*cmd, /\* optional command string \*/

struct regs\_t \*regs, /\* register to access \*/

struct mem\_t \*mem) /\* memory to access \*/

{

/\* just dump intermediate stats \*/

sim\_print\_stats(stream);

/\* no error \*/

return NULL;

}

/\* load program into simulated state \*/

void

sim\_load\_prog(char \*fname, /\* program to load \*/

int argc, char \*\*argv, /\* program arguments \*/

char \*\*envp) /\* program environment \*/

{

/\* load program text and data, set up environment, memory, and regs \*/

ld\_load\_prog(fname, argc, argv, envp, &regs, mem, TRUE);

/\* initialize the DLite debugger \*/

dlite\_init(md\_reg\_obj, dlite\_mem\_obj, cache\_mstate\_obj);

}

/\* print simulator-specific configuration information \*/

void

sim\_aux\_config(FILE \*stream) /\* output stream \*/

{

/\* nada \*/

}

/\* register simulator-specific statistics \*/

void

sim\_reg\_stats(struct stat\_sdb\_t \*sdb) /\* stats database \*/

{

int i;

/\* register baseline stats \*/

stat\_reg\_counter(sdb, "sim\_num\_insn",

"total number of instructions executed",

&sim\_num\_insn, sim\_num\_insn, NULL);

stat\_reg\_counter(sdb, "sim\_num\_refs",

"total number of loads and stores executed",

&sim\_num\_refs, 0, NULL);

stat\_reg\_int(sdb, "sim\_elapsed\_time",

"total simulation time in seconds",

&sim\_elapsed\_time, 0, NULL);

stat\_reg\_formula(sdb, "sim\_inst\_rate",

"simulation speed (in insts/sec)",

"sim\_num\_insn / sim\_elapsed\_time", NULL);

/\* register cache stats \*/

if (cache\_il1

&& (cache\_il1 != cache\_dl1 && cache\_il1 != cache\_dl2 && cache\_il1 != cache\_dl3 && cache\_il1 != cache\_dl4))

cache\_reg\_stats(cache\_il1, sdb);

if (cache\_il2

&& (cache\_il2 != cache\_dl1 && cache\_il2 != cache\_dl2 && cache\_il2 != cache\_dl3 && cache\_il2 != cache\_dl4))

cache\_reg\_stats(cache\_il2, sdb);

if (cache\_il3

&& (cache\_il3 != cache\_dl1 && cache\_il3 != cache\_dl2 && cache\_il3 != cache\_dl3 && cache\_il3 != cache\_dl4))

cache\_reg\_stats(cache\_il3, sdb);

if (cache\_il4

&& (cache\_il4 != cache\_dl1 && cache\_il4 != cache\_dl2 && cache\_il4 != cache\_dl3 && cache\_il4 != cache\_dl4))

cache\_reg\_stats(cache\_il4, sdb);

if (cache\_dl1)

cache\_reg\_stats(cache\_dl1, sdb);

if (cache\_dl2)

cache\_reg\_stats(cache\_dl2, sdb);

if (cache\_dl3)

cache\_reg\_stats(cache\_dl3, sdb);

if (cache\_dl4)

cache\_reg\_stats(cache\_dl4, sdb);

if (itlb)

cache\_reg\_stats(itlb, sdb);

if (dtlb)

cache\_reg\_stats(dtlb, sdb);

for (i=0; i<pcstat\_nelt; i++)

{

char buf[512], buf1[512];

struct stat\_stat\_t \*stat;

/\* track the named statistical variable by text address \*/

/\* find it... \*/

stat = stat\_find\_stat(sdb, pcstat\_vars[i]);

if (!stat)

fatal("cannot locate any statistic named `%s'", pcstat\_vars[i]);

/\* stat must be an integral type \*/

if (stat->sc != sc\_int && stat->sc != sc\_uint && stat->sc != sc\_counter)

fatal("`-pcstat' statistical variable `%s' is not an integral type",

stat->name);

/\* register this stat \*/

pcstat\_stats[i] = stat;

pcstat\_lastvals[i] = STATVAL(stat);

/\* declare the sparce text distribution \*/

sprintf(buf, "%s\_by\_pc", stat->name);

sprintf(buf1, "%s (by text address)", stat->desc);

pcstat\_sdists[i] = stat\_reg\_sdist(sdb, buf, buf1,

/\* initial value \*/0,

/\* print fmt \*/(PF\_COUNT|PF\_PDF),

/\* format \*/"0x%p %u %.2f",

/\* print fn \*/NULL);

}

ld\_reg\_stats(sdb);

mem\_reg\_stats(mem, sdb);

}

/\* dump simulator-specific auxiliary simulator statistics \*/

void

sim\_aux\_stats(FILE \*stream) /\* output stream \*/

{

/\* nada \*/

}

/\* un-initialize the simulator \*/

void

sim\_uninit(void)

{

/\* nada \*/

}

/\*

\* configure the execution engine

\*/

/\*

\* precise architected register accessors

\*/

/\* next program counter \*/

#define SET\_NPC(EXPR) (regs.regs\_NPC = (EXPR))

/\* current program counter \*/

#define CPC (regs.regs\_PC)

/\* general purpose registers \*/

#define GPR(N) (regs.regs\_R[N])

#define SET\_GPR(N,EXPR) (regs.regs\_R[N] = (EXPR))

#if defined(TARGET\_PISA)

/\* floating point registers, L->word, F->single-prec, D->double-prec \*/

#define FPR\_L(N) (regs.regs\_F.l[(N)])

#define SET\_FPR\_L(N,EXPR) (regs.regs\_F.l[(N)] = (EXPR))

#define FPR\_F(N) (regs.regs\_F.f[(N)])

#define SET\_FPR\_F(N,EXPR) (regs.regs\_F.f[(N)] = (EXPR))

#define FPR\_D(N) (regs.regs\_F.d[(N) >> 1])

#define SET\_FPR\_D(N,EXPR) (regs.regs\_F.d[(N) >> 1] = (EXPR))

/\* miscellaneous register accessors \*/

#define SET\_HI(EXPR) (regs.regs\_C.hi = (EXPR))

#define HI (regs.regs\_C.hi)

#define SET\_LO(EXPR) (regs.regs\_C.lo = (EXPR))

#define LO (regs.regs\_C.lo)

#define FCC (regs.regs\_C.fcc)

#define SET\_FCC(EXPR) (regs.regs\_C.fcc = (EXPR))

#elif defined(TARGET\_ALPHA)

/\* floating point registers, L->word, F->single-prec, D->double-prec \*/

#define FPR\_Q(N) (regs.regs\_F.q[N])

#define SET\_FPR\_Q(N,EXPR) (regs.regs\_F.q[N] = (EXPR))

#define FPR(N) (regs.regs\_F.d[N])

#define SET\_FPR(N,EXPR) (regs.regs\_F.d[N] = (EXPR))

/\* miscellaneous register accessors \*/

#define FPCR (regs.regs\_C.fpcr)

#define SET\_FPCR(EXPR) (regs.regs\_C.fpcr = (EXPR))

#define UNIQ (regs.regs\_C.uniq)

#define SET\_UNIQ(EXPR) (regs.regs\_C.uniq = (EXPR))

#else

#error No ISA target defined...

#endif

/\* precise architected memory state accessor macros \*/

#define \_\_READ\_CACHE(addr, SRC\_T) \

((dtlb \

? cache\_access(dtlb, Read, (addr), NULL, \

sizeof(SRC\_T), 0, NULL, NULL) \

: 0), \

(cache\_dl1 \

? cache\_access(cache\_dl1, Read, (addr), NULL, \

sizeof(SRC\_T), 0, NULL, NULL) \

: 0))

#define READ\_BYTE(SRC, FAULT) \

((FAULT) = md\_fault\_none, addr = (SRC), \

\_\_READ\_CACHE(addr, byte\_t), MEM\_READ\_BYTE(mem, addr))

#define READ\_HALF(SRC, FAULT) \

((FAULT) = md\_fault\_none, addr = (SRC), \

\_\_READ\_CACHE(addr, half\_t), MEM\_READ\_HALF(mem, addr))

#define READ\_WORD(SRC, FAULT) \

((FAULT) = md\_fault\_none, addr = (SRC), \

\_\_READ\_CACHE(addr, word\_t), MEM\_READ\_WORD(mem, addr))

#ifdef HOST\_HAS\_QWORD

#define READ\_QWORD(SRC, FAULT) \

((FAULT) = md\_fault\_none, addr = (SRC), \

\_\_READ\_CACHE(addr, qword\_t), MEM\_READ\_QWORD(mem, addr))

#endif /\* HOST\_HAS\_QWORD \*/

#define \_\_WRITE\_CACHE(addr, DST\_T) \

((dtlb \

? cache\_access(dtlb, Write, (addr), NULL, \

sizeof(DST\_T), 0, NULL, NULL) \

: 0), \

(cache\_dl1 \

? cache\_access(cache\_dl1, Write, (addr), NULL, \

sizeof(DST\_T), 0, NULL, NULL) \

: 0))

#define WRITE\_BYTE(SRC, DST, FAULT) \

((FAULT) = md\_fault\_none, addr = (DST), \

\_\_WRITE\_CACHE(addr, byte\_t), MEM\_WRITE\_BYTE(mem, addr, (SRC)))

#define WRITE\_HALF(SRC, DST, FAULT) \

((FAULT) = md\_fault\_none, addr = (DST), \

\_\_WRITE\_CACHE(addr, half\_t), MEM\_WRITE\_HALF(mem, addr, (SRC)))

#define WRITE\_WORD(SRC, DST, FAULT) \

((FAULT) = md\_fault\_none, addr = (DST), \

\_\_WRITE\_CACHE(addr, word\_t), MEM\_WRITE\_WORD(mem, addr, (SRC)))

#ifdef HOST\_HAS\_QWORD

#define WRITE\_QWORD(SRC, DST, FAULT) \

((FAULT) = md\_fault\_none, addr = (DST), \

\_\_WRITE\_CACHE(addr, qword\_t), MEM\_WRITE\_QWORD(mem, addr, (SRC)))

#endif /\* HOST\_HAS\_QWORD \*/

/\* system call memory access function \*/

enum md\_fault\_type

dcache\_access\_fn(struct mem\_t \*mem, /\* memory space to access \*/

enum mem\_cmd cmd, /\* memory access cmd, Read or Write \*/

md\_addr\_t addr, /\* data address to access \*/

void \*p, /\* data input/output buffer \*/

int nbytes) /\* number of bytes to access \*/

{

if (dtlb)

cache\_access(dtlb, cmd, addr, NULL, nbytes, 0, NULL, NULL);

if (cache\_dl1)

cache\_access(cache\_dl1, cmd, addr, NULL, nbytes, 0, NULL, NULL);

return mem\_access(mem, cmd, addr, p, nbytes);

}

/\* system call handler macro \*/

#define SYSCALL(INST) \

(flush\_on\_syscalls \

? ((dtlb ? cache\_flush(dtlb, 0) : 0), \

(cache\_dl1 ? cache\_flush(cache\_dl1, 0) : 0), \

(cache\_dl2 ? cache\_flush(cache\_dl2, 0) : 0), \

sys\_syscall(&regs, mem\_access, mem, INST, TRUE)) \

: sys\_syscall(&regs, dcache\_access\_fn, mem, INST, TRUE))

/\* start simulation, program loaded, processor precise state initialized \*/

void

sim\_main(void)

{

int i;

md\_inst\_t inst;

register md\_addr\_t addr;

enum md\_opcode op;

register int is\_write;

enum md\_fault\_type fault;

fprintf(stderr, "sim: \*\* starting functional simulation w/ caches \*\*\n");

/\* set up initial default next PC \*/

regs.regs\_NPC = regs.regs\_PC + sizeof(md\_inst\_t);

/\* check for DLite debugger entry condition \*/

if (dlite\_check\_break(regs.regs\_PC, /\* no access \*/0, /\* addr \*/0, 0, 0))

dlite\_main(regs.regs\_PC - sizeof(md\_inst\_t), regs.regs\_PC,

sim\_num\_insn, &regs, mem);

while (TRUE)

{

/\* maintain $r0 semantics \*/

regs.regs\_R[MD\_REG\_ZERO] = 0;

#ifdef TARGET\_ALPHA

regs.regs\_F.d[MD\_REG\_ZERO] = 0.0;

#endif /\* TARGET\_ALPHA \*/

/\* get the next instruction to execute \*/

if (itlb)

cache\_access(itlb, Read, IACOMPRESS(regs.regs\_PC),

NULL, ISCOMPRESS(sizeof(md\_inst\_t)), 0, NULL, NULL);

if (cache\_il1)

cache\_access(cache\_il1, Read, IACOMPRESS(regs.regs\_PC),

NULL, ISCOMPRESS(sizeof(md\_inst\_t)), 0, NULL, NULL);

MD\_FETCH\_INST(inst, mem, regs.regs\_PC);

/\* keep an instruction count \*/

sim\_num\_insn++;

/\* set default reference address and access mode \*/

addr = 0; is\_write = FALSE;

/\* set default fault - none \*/

fault = md\_fault\_none;

/\* decode the instruction \*/

MD\_SET\_OPCODE(op, inst);

/\* execute the instruction \*/

switch (op)

{

#define DEFINST(OP,MSK,NAME,OPFORM,RES,FLAGS,O1,O2,I1,I2,I3) \

case OP: \

SYMCAT(OP,\_IMPL); \

break;

#define DEFLINK(OP,MSK,NAME,MASK,SHIFT) \

case OP: \

panic("attempted to execute a linking opcode");

#define CONNECT(OP)

#define DECLARE\_FAULT(FAULT) \

{ fault = (FAULT); break; }

#include "machine.def"

default:

panic("attempted to execute a bogus opcode");

}

if (fault != md\_fault\_none)

fatal("fault (%d) detected @ 0x%08p", fault, regs.regs\_PC);

if (MD\_OP\_FLAGS(op) & F\_MEM)

{

sim\_num\_refs++;

if (MD\_OP\_FLAGS(op) & F\_STORE)

is\_write = TRUE;

}

/\* update any stats tracked by PC \*/

for (i=0; i < pcstat\_nelt; i++)

{

counter\_t newval;

int delta;

/\* check if any tracked stats changed \*/

newval = STATVAL(pcstat\_stats[i]);

delta = newval - pcstat\_lastvals[i];

if (delta != 0)

{

stat\_add\_samples(pcstat\_sdists[i], regs.regs\_PC, delta);

pcstat\_lastvals[i] = newval;

}

}

/\* check for DLite debugger entry condition \*/

if (dlite\_check\_break(regs.regs\_NPC,

is\_write ? ACCESS\_WRITE : ACCESS\_READ,

addr, sim\_num\_insn, sim\_num\_insn))

dlite\_main(regs.regs\_PC, regs.regs\_NPC, sim\_num\_insn, &regs, mem);

/\* go to the next instruction \*/

regs.regs\_PC = regs.regs\_NPC;

regs.regs\_NPC += sizeof(md\_inst\_t);

/\* finish early? \*/

if (max\_insts && sim\_num\_insn >= max\_insts)

return;

}

}

The results of cache\_4a.out are below.

sim-cache: SimpleScalar/PISA Tool Set version 3.0 of August, 2003.

Copyright (c) 1994-2003 by Todd M. Austin, Ph.D. and SimpleScalar, LLC.

All Rights Reserved. This version of SimpleScalar is licensed for academic

non-commercial use. No portion of this work may be used by any commercial

entity, or for any commercial purpose, without the prior written permission

of SimpleScalar, LLC (info@simplescalar.com).

sim: command line: ./sim-cache -config ../../../Downloads/sim-cache-l3-master/lab\_config/cache\_4a.cfg -redir:sim cache\_4a.out ./tests-pisa/bin.little/test-math

sim: simulation started @ Mon Nov 29 09:19:49 2021, options follow:

sim-cache: This simulator implements a functional cache simulator. Cache

statistics are generated for a user-selected cache and TLB configuration,

which may include up to four levels of instruction and data cache (with any

levels unified), and one level of instruction and data TLBs. No timing

information is generated.

# -config # load configuration from a file

# -dumpconfig # dump configuration to a file

# -h false # print help message

# -v false # verbose operation

# -d false # enable debug message

# -i false # start in Dlite debugger

-seed 1 # random number generator seed (0 for timer seed)

# -q false # initialize and terminate immediately

# -chkpt <null> # restore EIO trace execution from <fname>

# -redir:sim cache\_4a.out # redirect simulator output to file (non-interactive only)

# -redir:prog <null> # redirect simulated program output to file

-nice 0 # simulator scheduling priority

-max:inst 0 # maximum number of inst's to execute

-cache:dl1 dl1:64:64:1:l # l1 data cache config, i.e., {<config>|none}

-cache:dl2 dl2:128:64:1:l # l2 data cache config, i.e., {<config>|none}

-cache:dl3 dl3:256:64:1:l # l3 data cache config, i.e., {<config>|none}

-cache:dl4 dl4:512:64:1:l # l4 data cache config, i.e., {<config>|none}

-cache:il1 il1:64:64:1:l # l1 inst cache config, i.e., {<config>|dl1|dl2|dl3|dl4|none}

-cache:il2 il2:128:64:1:l; # l2 instruction cache config, i.e., {<config>|dl2|dl3|dl4|none}

-cache:il3 il3:256:64:1:l; # l3 instruction cache config, i.e., {<config>|dl3|dl4|none}

-cache:il4 il4:512:64:1:l; # l4 instruction cache config, i.e., {<config>|dl4|none}

-tlb:itlb none # instruction TLB config, i.e., {<config>|none}

-tlb:dtlb none # data TLB config, i.e., {<config>|none}

-flush false # flush caches on system calls

-cache:icompress false # convert 64-bit inst addresses to 32-bit inst equivalents

# -pcstat <null> # profile stat(s) against text addr's (mult uses ok)

The cache config parameter <config> has the following format:

<name>:<nsets>:<bsize>:<assoc>:<repl>

<name> - name of the cache being defined

<nsets> - number of sets in the cache

<bsize> - block size of the cache

<assoc> - associativity of the cache

<repl> - block replacement strategy, 'l'-LRU, 'f'-FIFO, 'r'-random

Examples: -cache:dl1 dl1:4096:32:1:l

-dtlb dtlb:128:4096:32:r

Cache levels can be unified by pointing a level of the instruction cache

hierarchy at the data cache hiearchy using the "dl1" and "dl2" cache

configuration arguments. Most sensible combinations are supported, e.g.,

A unified l2 cache (il2 is pointed at dl2):

-cache:il1 il1:128:64:1:l -cache:il2 dl2

-cache:dl1 dl1:256:32:1:l -cache:dl2 ul2:1024:64:2:l

Or, a fully unified cache hierarchy (il1 pointed at dl1):

-cache:il1 dl1

-cache:dl1 ul1:256:32:1:l -cache:dl2 ul2:1024:64:2:l

sim: \*\* starting functional simulation w/ caches \*\*

sim: \*\* simulation statistics \*\*

sim\_num\_insn 213745 # total number of instructions executed

sim\_num\_refs 56902 # total number of loads and stores executed

sim\_elapsed\_time 1 # total simulation time in seconds

sim\_inst\_rate 213745.0000 # simulation speed (in insts/sec)

il1.accesses 213745 # total number of accesses

il1.hits 193259 # total number of hits

il1.misses 20486 # total number of misses

il1.replacements 20422 # total number of replacements

il1.writebacks 0 # total number of writebacks

il1.invalidations 0 # total number of invalidations

il1.miss\_rate 0.0958 # miss rate (i.e., misses/ref)

il1.repl\_rate 0.0955 # replacement rate (i.e., repls/ref)

il1.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

il1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

il2.accesses 20486 # total number of accesses

il2.hits 5045 # total number of hits

il2.misses 15441 # total number of misses

il2.replacements 15313 # total number of replacements

il2.writebacks 0 # total number of writebacks

il2.invalidations 0 # total number of invalidations

il2.miss\_rate 0.7537 # miss rate (i.e., misses/ref)

il2.repl\_rate 0.7475 # replacement rate (i.e., repls/ref)

il2.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

il2.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

il3.accesses 15441 # total number of accesses

il3.hits 4910 # total number of hits

il3.misses 10531 # total number of misses

il3.replacements 10275 # total number of replacements

il3.writebacks 0 # total number of writebacks

il3.invalidations 0 # total number of invalidations

il3.miss\_rate 0.6820 # miss rate (i.e., misses/ref)

il3.repl\_rate 0.6654 # replacement rate (i.e., repls/ref)

il3.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

il3.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

il4.accesses 10531 # total number of accesses

il4.hits 6141 # total number of hits

il4.misses 4390 # total number of misses

il4.replacements 3893 # total number of replacements

il4.writebacks 0 # total number of writebacks

il4.invalidations 0 # total number of invalidations

il4.miss\_rate 0.4169 # miss rate (i.e., misses/ref)

il4.repl\_rate 0.3697 # replacement rate (i.e., repls/ref)

il4.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

il4.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dl1.accesses 57483 # total number of accesses

dl1.hits 56364 # total number of hits

dl1.misses 1119 # total number of misses

dl1.replacements 1055 # total number of replacements

dl1.writebacks 625 # total number of writebacks

dl1.invalidations 0 # total number of invalidations

dl1.miss\_rate 0.0195 # miss rate (i.e., misses/ref)

dl1.repl\_rate 0.0184 # replacement rate (i.e., repls/ref)

dl1.wb\_rate 0.0109 # writeback rate (i.e., wrbks/ref)

dl1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dl2.accesses 1744 # total number of accesses

dl2.hits 1120 # total number of hits

dl2.misses 624 # total number of misses

dl2.replacements 496 # total number of replacements

dl2.writebacks 326 # total number of writebacks

dl2.invalidations 0 # total number of invalidations

dl2.miss\_rate 0.3578 # miss rate (i.e., misses/ref)

dl2.repl\_rate 0.2844 # replacement rate (i.e., repls/ref)

dl2.wb\_rate 0.1869 # writeback rate (i.e., wrbks/ref)

dl2.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dl3.accesses 950 # total number of accesses

dl3.hits 637 # total number of hits

dl3.misses 313 # total number of misses

dl3.replacements 97 # total number of replacements

dl3.writebacks 86 # total number of writebacks

dl3.invalidations 0 # total number of invalidations

dl3.miss\_rate 0.3295 # miss rate (i.e., misses/ref)

dl3.repl\_rate 0.1021 # replacement rate (i.e., repls/ref)

dl3.wb\_rate 0.0905 # writeback rate (i.e., wrbks/ref)

dl3.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dl4.accesses 399 # total number of accesses

dl4.hits 101 # total number of hits

dl4.misses 298 # total number of misses

dl4.replacements 12 # total number of replacements

dl4.writebacks 9 # total number of writebacks

dl4.invalidations 0 # total number of invalidations

dl4.miss\_rate 0.7469 # miss rate (i.e., misses/ref)

dl4.repl\_rate 0.0301 # replacement rate (i.e., repls/ref)

dl4.wb\_rate 0.0226 # writeback rate (i.e., wrbks/ref)

dl4.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

ld\_text\_base 0x00400000 # program text (code) segment base

ld\_text\_size 91744 # program text (code) size in bytes

ld\_data\_base 0x10000000 # program initialized data segment base

ld\_data\_size 13028 # program init'ed `.data' and uninit'ed `.bss' size in bytes

ld\_stack\_base 0x7fffc000 # program stack segment base (highest address in stack)

ld\_stack\_size 16384 # program initial stack size

ld\_prog\_entry 0x00400140 # program entry point (initial PC)

ld\_environ\_base 0x7fff8000 # program environment base address address

ld\_target\_big\_endian 0 # target executable endian-ness, non-zero if big endian

mem.page\_count 33 # total number of pages allocated

mem.page\_mem 132k # total size of memory pages allocated

mem.ptab\_misses 34 # total first level page table misses

mem.ptab\_accesses 1548145 # total page table accesses

mem.ptab\_miss\_rate 0.0000 # first level page table miss rate

The results of cache\_4b.out are below.

sim-cache: SimpleScalar/PISA Tool Set version 3.0 of August, 2003.

Copyright (c) 1994-2003 by Todd M. Austin, Ph.D. and SimpleScalar, LLC.

All Rights Reserved. This version of SimpleScalar is licensed for academic

non-commercial use. No portion of this work may be used by any commercial

entity, or for any commercial purpose, without the prior written permission

of SimpleScalar, LLC (info@simplescalar.com).

sim: command line: ./sim-cache -config ../../../Downloads/sim-cache-l3-master/lab\_config/cache\_4b.cfg -redir:sim cache\_4b.out ./tests-pisa/bin.little/test-math

sim: simulation started @ Mon Nov 29 09:20:00 2021, options follow:

sim-cache: This simulator implements a functional cache simulator. Cache

statistics are generated for a user-selected cache and TLB configuration,

which may include up to four levels of instruction and data cache (with any

levels unified), and one level of instruction and data TLBs. No timing

information is generated.

# -config # load configuration from a file

# -dumpconfig # dump configuration to a file

# -h false # print help message

# -v false # verbose operation

# -d false # enable debug message

# -i false # start in Dlite debugger

-seed 1 # random number generator seed (0 for timer seed)

# -q false # initialize and terminate immediately

# -chkpt <null> # restore EIO trace execution from <fname>

# -redir:sim cache\_4b.out # redirect simulator output to file (non-interactive only)

# -redir:prog <null> # redirect simulated program output to file

-nice 0 # simulator scheduling priority

-max:inst 0 # maximum number of inst's to execute

-cache:dl1 dl1:64:64:1:l # l1 data cache config, i.e., {<config>|none}

-cache:dl2 dl2:128:64:1:l # l2 data cache config, i.e., {<config>|none}

-cache:dl3 dl3:256:64:1:l # l3 data cache config, i.e., {<config>|none}

-cache:dl4 dl4:1024:64:1:l # l4 data cache config, i.e., {<config>|none}

-cache:il1 il1:64:64:1:l # l1 inst cache config, i.e., {<config>|dl1|dl2|dl3|dl4|none}

-cache:il2 il2:128:64:1:l; # l2 instruction cache config, i.e., {<config>|dl2|dl3|dl4|none}

-cache:il3 il3:256:64:1:l; # l3 instruction cache config, i.e., {<config>|dl3|dl4|none}

-cache:il4 dl4 # l4 instruction cache config, i.e., {<config>|dl4|none}

-tlb:itlb none # instruction TLB config, i.e., {<config>|none}

-tlb:dtlb none # data TLB config, i.e., {<config>|none}

-flush false # flush caches on system calls

-cache:icompress false # convert 64-bit inst addresses to 32-bit inst equivalents

# -pcstat <null> # profile stat(s) against text addr's (mult uses ok)

The cache config parameter <config> has the following format:

<name>:<nsets>:<bsize>:<assoc>:<repl>

<name> - name of the cache being defined

<nsets> - number of sets in the cache

<bsize> - block size of the cache

<assoc> - associativity of the cache

<repl> - block replacement strategy, 'l'-LRU, 'f'-FIFO, 'r'-random

Examples: -cache:dl1 dl1:4096:32:1:l

-dtlb dtlb:128:4096:32:r

Cache levels can be unified by pointing a level of the instruction cache

hierarchy at the data cache hiearchy using the "dl1" and "dl2" cache

configuration arguments. Most sensible combinations are supported, e.g.,

A unified l2 cache (il2 is pointed at dl2):

-cache:il1 il1:128:64:1:l -cache:il2 dl2

-cache:dl1 dl1:256:32:1:l -cache:dl2 ul2:1024:64:2:l

Or, a fully unified cache hierarchy (il1 pointed at dl1):

-cache:il1 dl1

-cache:dl1 ul1:256:32:1:l -cache:dl2 ul2:1024:64:2:l

sim: \*\* starting functional simulation w/ caches \*\*

sim: \*\* simulation statistics \*\*

sim\_num\_insn 213745 # total number of instructions executed

sim\_num\_refs 56902 # total number of loads and stores executed

sim\_elapsed\_time 1 # total simulation time in seconds

sim\_inst\_rate 213745.0000 # simulation speed (in insts/sec)

il1.accesses 213745 # total number of accesses

il1.hits 193259 # total number of hits

il1.misses 20486 # total number of misses

il1.replacements 20422 # total number of replacements

il1.writebacks 0 # total number of writebacks

il1.invalidations 0 # total number of invalidations

il1.miss\_rate 0.0958 # miss rate (i.e., misses/ref)

il1.repl\_rate 0.0955 # replacement rate (i.e., repls/ref)

il1.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

il1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

il2.accesses 20486 # total number of accesses

il2.hits 5045 # total number of hits

il2.misses 15441 # total number of misses

il2.replacements 15313 # total number of replacements

il2.writebacks 0 # total number of writebacks

il2.invalidations 0 # total number of invalidations

il2.miss\_rate 0.7537 # miss rate (i.e., misses/ref)

il2.repl\_rate 0.7475 # replacement rate (i.e., repls/ref)

il2.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

il2.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

il3.accesses 15441 # total number of accesses

il3.hits 4910 # total number of hits

il3.misses 10531 # total number of misses

il3.replacements 10275 # total number of replacements

il3.writebacks 0 # total number of writebacks

il3.invalidations 0 # total number of invalidations

il3.miss\_rate 0.6820 # miss rate (i.e., misses/ref)

il3.repl\_rate 0.6654 # replacement rate (i.e., repls/ref)

il3.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

il3.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dl1.accesses 57483 # total number of accesses

dl1.hits 56364 # total number of hits

dl1.misses 1119 # total number of misses

dl1.replacements 1055 # total number of replacements

dl1.writebacks 625 # total number of writebacks

dl1.invalidations 0 # total number of invalidations

dl1.miss\_rate 0.0195 # miss rate (i.e., misses/ref)

dl1.repl\_rate 0.0184 # replacement rate (i.e., repls/ref)

dl1.wb\_rate 0.0109 # writeback rate (i.e., wrbks/ref)

dl1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dl2.accesses 1744 # total number of accesses

dl2.hits 1120 # total number of hits

dl2.misses 624 # total number of misses

dl2.replacements 496 # total number of replacements

dl2.writebacks 326 # total number of writebacks

dl2.invalidations 0 # total number of invalidations

dl2.miss\_rate 0.3578 # miss rate (i.e., misses/ref)

dl2.repl\_rate 0.2844 # replacement rate (i.e., repls/ref)

dl2.wb\_rate 0.1869 # writeback rate (i.e., wrbks/ref)

dl2.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dl3.accesses 950 # total number of accesses

dl3.hits 637 # total number of hits

dl3.misses 313 # total number of misses

dl3.replacements 97 # total number of replacements

dl3.writebacks 86 # total number of writebacks

dl3.invalidations 0 # total number of invalidations

dl3.miss\_rate 0.3295 # miss rate (i.e., misses/ref)

dl3.repl\_rate 0.1021 # replacement rate (i.e., repls/ref)

dl3.wb\_rate 0.0905 # writeback rate (i.e., wrbks/ref)

dl3.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dl4.accesses 10930 # total number of accesses

dl4.hits 8982 # total number of hits

dl4.misses 1948 # total number of misses

dl4.replacements 1169 # total number of replacements

dl4.writebacks 55 # total number of writebacks

dl4.invalidations 0 # total number of invalidations

dl4.miss\_rate 0.1782 # miss rate (i.e., misses/ref)

dl4.repl\_rate 0.1070 # replacement rate (i.e., repls/ref)

dl4.wb\_rate 0.0050 # writeback rate (i.e., wrbks/ref)

dl4.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

ld\_text\_base 0x00400000 # program text (code) segment base

ld\_text\_size 91744 # program text (code) size in bytes

ld\_data\_base 0x10000000 # program initialized data segment base

ld\_data\_size 13028 # program init'ed `.data' and uninit'ed `.bss' size in bytes

ld\_stack\_base 0x7fffc000 # program stack segment base (highest address in stack)

ld\_stack\_size 16384 # program initial stack size

ld\_prog\_entry 0x00400140 # program entry point (initial PC)

ld\_environ\_base 0x7fff8000 # program environment base address address

ld\_target\_big\_endian 0 # target executable endian-ness, non-zero if big endian

mem.page\_count 33 # total number of pages allocated

mem.page\_mem 132k # total size of memory pages allocated

mem.ptab\_misses 34 # total first level page table misses

mem.ptab\_accesses 1548145 # total page table accesses

mem.ptab\_miss\_rate 0.0000 # first level page table miss rate