Homework 4

CIS-655 ADVANCED COMPUTER ARCHITECTURE

PROF. MOHAMMED ABDALLAH

11/29/2021

Anthony Redamonti
Jaime L Ponicki
Ang Li
SYRACUSE UNIVERSITY

A. 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.

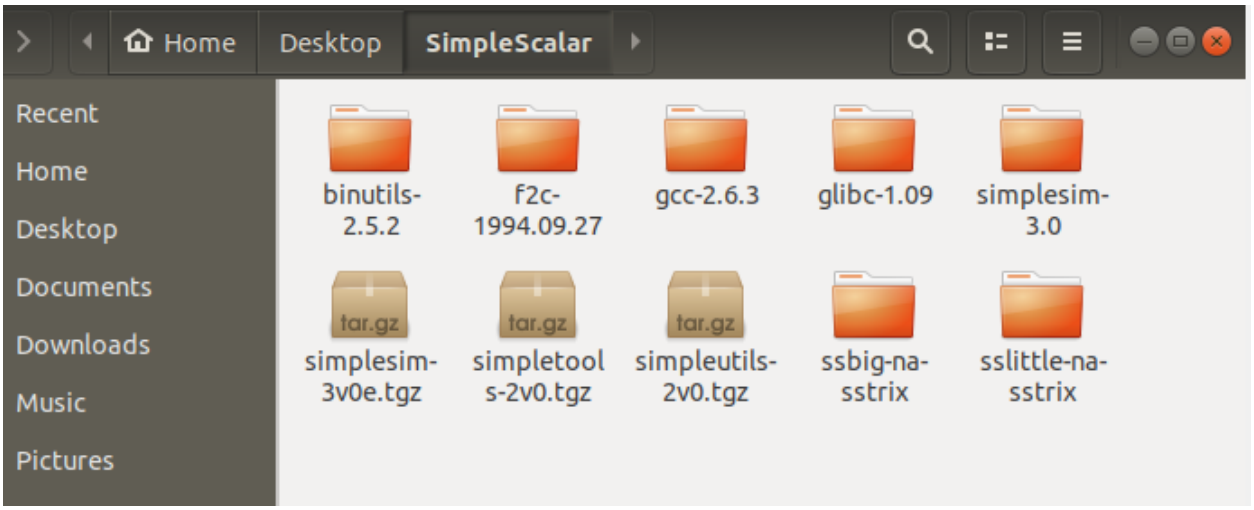
Create a folder in the Desktop named “SimpleScalar” and copy the simplesim-3v0e.tgz, simpletools-2v0.tgz, and simpleutils-2v0.tgz into the folder. Unpack each of them with the following commands:

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=../

```
anthonyredamonti@anthonyredamonti-VirtualBox:~/Desktop/SimpleScalar$ cd binutil
s-2.5.2/
anthonyredamonti@anthonyredamonti-VirtualBox:~/Desktop/SimpleScalar/binutils-2.
5.2$ ./configure --host=i386-*-gnu/linux --target=ssbig-na-sstrix --with-gnu-as
 --with-gnu-ld --prefix=../
Created "Makefile" in /home/anthonyredamonti/Desktop/SimpleScalar/binutils-2.5.
2
anthonyredamonti@anthonyredamonti-VirtualBox:~/Desktop/SimpleScalar/binutils-2.
5.2$ make
make[1]: Entering directory '/home/anthonyredamonti/Desktop/SimpleScalar/binuti
ls-2.5.2/libiberty'
echo "# !Automatically generated from ./functions.def"\
  "- DO NOT EDIT!" >needed2.awk
grep '^DEFVAR(' < ./functions.def \
 | sed -e '/DEFVAR/s|DEFVAR.\([^,]*\).*|/\1/ { printf "#ifndef NEED_\1\\n#defin
e NEED \1\\n#endif\\n" }|' \
```

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

```
anthonyredamonti@anthonyredamonti-VirtualBox:~/Desktop/SimpleScalar/binutils-2.
5.2$ make install
Making ../...
Making ../...
Making ..//ssbig-na-sstrix...
make[1]: Entering directory '/home/anthonyredamonti/Desktop/SimpleScalar/binuti
ls-2.5.2/bfd'
/home/anthonyredamonti/Desktop/SimpleScalar/binutils-2.5.2/install.sh -c -m 644
 libbfd.a ..//lib/libbfd.a
install:  libbfd.a does not exist
Makefile:456: recipe for target 'install' failed
make[1]: *** [install] Error 1
make[1]: Leaving directory '/home/anthonyredamonti/Desktop/SimpleScalar/binutil
s-2.5.2/bfd'
```

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

cd ../simplesim-3.0/

make

```
anthonyredamonti@anthonyredamonti-VirtualBox:~/Desktop/SimpleScalar/binutils-2.
5.2$ cd ../simplesim-3.0/
anthonyredamonti@anthonyredamonti-VirtualBox:~/Desktop/SimpleScalar/simplesim-3
.0$ make
gcc -DDEBUG -o sysprobe sysprobe.c
sysprobe.c: In function 'main':
sysprobe.c:263:39: warning: format '%d' expects argument of type 'int', but arg
ument 3 has type 'long unsigned int' [-Wformat=]
       fprintf(stdout, "sizeof(int) = %d\n", sizeof(int));
                                       ~^
                                       %ld
sysprobe.c:264:40: warning: format '%d' expects argument of type 'int', but arg
ument 3 has type 'long unsigned int' [-Wformat=]
       fprintf(stdout, "sizeof(long) = %d\n", sizeof(long));
                                        ~^
                                        %ld
endian probe results: little
probe flags: -DBYTES_LITTLE_ENDIAN -DWORDS_LITTLE_ENDIAN -DFAST_SRL -DFAST_SRA
-DGZIP_PATH="/bin/gzip"
probe libs: -lm
gcc `./sysprobe -flags` -DDEBUG -O0 -g -Wall    -c sim-fast.c
gcc `./sysprobe -flags` -DDEBUG -O0 -g -Wall    -c main.c
gcc `./sysprobe -flags` -DDEBUG -O0 -g -Wall    -c syscall.c
```

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=../

```
anthonyredamonti@anthonyredamonti-VirtualBox:~/Desktop/SimpleScalar/simplesim-3
.0$ cd ../gcc-2.6.3/
anthonyredamonti@anthonyredamonti-VirtualBox:~/Desktop/SimpleScalar/gcc-2.6.3$
./configure --host=i386-*-gnu/linux --target=ssbig-na-sstrix --with-gnu-as --wi
th-gnu-ld --prefix=../
Linked `config.h' to `./config/i386/xm-linux.h'
Linked `tm.h' to `./config/ss/ssbig.h'
Linked `aux-output.c' to `./config/ss/ss.c'
Linked `tconfig.h' to `./config/ss/xm-ss.h'
Linked `hconfig.h' to `./config/i386/xm-linux.h'
Linked `md' to `./config/ss/ss.md'
Merged i386/x-linux.
Merged ss/t-ss-gas.
Merged c++ fragment(s).
Created `./Makefile'.
Merged i386/x-linux.
Merged ss/t-ss-gas.
Created `cp/Makefile'.
Links are now set up to build a cross-compiler for ssbig-na-sstrix
  from i386-*-linux.
```

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

```
anthonyredamonti@anthonyredamonti-VirtualBox:~/Desktop/SimpleScalar/gcc-2.6.3$
make LANGUAGES=c
gcc -DCROSS_COMPILE -DIN_GCC -DPOSIX  -g     -I. -I. -I./config \
  -DGCC_INCLUDE_DIR=\"..//lib/gcc-lib/ssbig-na-sstrix/2.6.3/include\" \
  -DGPLUSPLUS_INCLUDE_DIR=\"..//lib/g++-include\" \
  -DLOCAL_INCLUDE_DIR=\"/usr/local/include\" \
  -DCROSS_INCLUDE_DIR=\"..//lib/gcc-lib/ssbig-na-sstrix/2.6.3/sys-include\" \
  -DTOOL_INCLUDE_DIR=\"..//ssbig-na-sstrix/include\" \
  -c `echo ./cccp.c | sed 's,^\./,,'`
cccp.c:194:14: error: conflicting types for ‘sys_errlist’
 extern char *sys_errlist[];
              ^~~~~~~~~~~
In file included from /usr/include/stdio.h:781:0,
                 from cccp.c:77:
/usr/include/x86_64-linux-gnu/bits/sys_errlist.h:27:26: note: previous declarat
ion of ‘sys_errlist’ was here
 extern const char *const sys_errlist[];
                          ^~~~~~~~~~~
cccp.c:276:8: warning: type defaults to ‘int’ in declaration of ‘is_system_incl
ude’ [-Wimplicit-int]
 static is_system_include ();
```

The installation is complete.

B. 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:

```
anthonyredamonti@anthonyredamonti-VirtualBox:~$ ./sim-cache -config ../../../Do
wnloads/sim-cache-l3-master/lab_config/cache_4a.cfg -redir:sim cache_4a.out ./t
ests-pisa/bin.little/test-math
```

```
anthonyredamonti@anthonyredamonti-VirtualBox:~$ ./sim-cache -config ../../../Do
wnloads/sim-cache-l3-master/lab_config/cache_4b.cfg -redir:sim cache_4b.out ./t
ests-pisa/bin.little/test-math
```

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.

C. 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
```