HW#2
COEN-4730 Computer Architecture

Objective

To learn about Dinero IV - a cache simulator for memory reference traces.
Use Dinero simulator to gain insights into replacement policy, unified vs. split, and multi-level caches.

1. Prerequisite: Linux in a Virtual Machine

If you have already a Linux machine or you have already done this before, you do not need to do this step.

The term virtualization means that you can have another OS over an existing OS. For instance, you can run Windows on a Mac or you may install Linux on a Windows 7 machine using virtualization software.

Now, create an Ubuntu VM following the steps described in this tutorial:
https://henricasanova.github.io/files/vbox/VirtualBoxUbuntuHowTo.html
Note: Right after installing the VirtualBox, select the Settings as appropriately for your machine as described here:
http://www.sevenforums.com/tutorials/130922-linux-install-windows-7-virtual-machine-using-virtualbox.html
(see Part 1) and then continue with the Ubuntu's installation as described in the Hawaii tutorial.

At this time you should have Ubuntu installed.

Some basics:

Launch Ubuntu and open a new terminal.

Create for example a new folder in your home directory:
```
$ mkdir multicore
```
To search for something that you may want to install, you can use "apt-cache search"; for example:
```
$ sudo apt-cache search emacs
```
To install something from what you found, you can use "apt-get install"; for example:
```
$ sudo apt-get install build-essential
```
As other examples, to install for example svn, flex, bison, and emacs you could do:
```
$ sudo apt-get install subversion
$ sudo apt-get install flex
$ sudo apt-get install bison
$ sudo apt-get install emacs23
```

2. DINERO IV Trace-driven Uniprocessor Simulator

Note: This short description is from the webpage of the simulator.

Dinero IV is a cache simulator for memory reference traces.

Some deep-seated limitations:
- Dinero IV is not a timing simulator. There is no notion of simulated time or cycles, only references.

- Dinero IV is not a functional simulator. Data & instructions do not move in and out of the caches; in fact they don't exist! **The primary result of simulation with Dinero IV is hit and miss information.**
- Dinero IV isn't multi-threaded. If you have a multiprocessor with enough memory, you can run multiple independent simulations concurrently.

The basic idea is to simulate a memory hierarchy consisting of various caches connected as one or more trees, with reference sources (the processors) at the leaves and a memory at each root. The various parameters of each cache can be set separately (architecture, policy, statistics). During initialization, the configuration to be simulated is built up, one cache at a time, starting with each memory as a special case. After initialization, each reference is fed to the appropriate top-level cache by a single simple function call. Lower levels of the hierarchy are handled automatically.

Preparation

Download and install Dinero IV. Take some time to browse its documentation and take a quick look at the source code (to just have an idea about how it was programmed). Installation Steps:

Step 1
Download Dinero IV (d4-7.tar.gz) from the following website and place the file in a directory of choice:
http://pages.cs.wisc.edu/~markhill/DineroIV/

Step 2
Unzip and untar the downloaded by running the following commands in the terminal window:
> gunzip d4-7.tar.gz
> tar -xvf d4-7.tar

Step 3
Inside a terminal window, cd to the d4-7 folder that was created in Step 2 and compile dinero with the following commands:
> cd d4-7
> ./configure
> make
This creates the executable dineroIV.

Step 4 (Traces)
You will not have to generate your own trace files; that has been done. Traces for three programs, cc1 (C compiler), spice (a circuit simulator) and tex (a document formatter) are provided as part of the archive for this assignment; they are in compressed form. To use them, copy them into a directory you have created for this project and uncompress them. For example, to uncompress the cc1 trace file, you should do:
> gunzip cc1.din.Z
This way you get cc1.din trace file. The names of all three trace files are cc1.din, spice.din, and tex.din respectively.

Step 5 (Testing)
Inside the d4-7/ folder is the dineroIV executable, which is the dinero program. The way it's run is like this:
> ./dineroIV (options) < trace_file_name

Here options can specify the cache size, cache block size, and cache layout options; trace_file_name is the name of the trace file.
To check out the available options:

> ./dineroIV -help

You should see something like this:

```
Usage: dineroIV [options]
Valid options:
 -help              Print this help message
 -copyright         Give details on copyright and lack of warranty
 -contact           Where to get the latest version or contact the authors
 -dineroIII         Explain replacements for Dinero III options
 -custom F          Generate and run custom simulator named F
 -lN-Tsize P        Size
 -lN-Tbsize P       Block size
 -lN-Tsbsize P      Sub-block size (default same as block size)
 -lN-Tassoc U       Associativity (default 1)
 -lN-Trepl C        Replacement policy
                    (l=LRU, f=FIFO, r=random) (default l)
 -lN-Tfetch C       Fetch policy
                    (d=demand, a=always, m=miss, t=tagged,
                     l=load forward, s=subblock) (default d)
 -lN-Tpfdist U      Prefetch distance (in sub-blocks) (default 1)
 -lN-Tpfabort U     Prefetch abort percentage (0-100) (default 0)
 -lN-Twalloc C      Write allocate policy
                    (a=always, n=never, f=nofetch) (default a)
 -lN-Twback C       Write back policy
                    (a=always, n=never, f=nofetch) (default a)
 -lN-Tccc           Compulsory/Capacity/Conflict miss statistics
 -skipcount U       Skip initial U references
 -flushcount U      Flush cache every U references
 -maxcount U        Stop simulation after U references
 -stat-interval U   Show statistics after every U references
 -informat C        Input trace format
                    (D=extended din, d=traditional din, p=pixie32, P=pixie64,
                    b=binary) (default D)
 -on-trigger A      Trigger address to start simulation
 -off-trigger A     Trigger address to stop simulation
 -stat-idcombine    Combine I&D cache stats
Key:
 U unsigned decimal integer
 S like U but with optional [kKmMgG] scaling suffix
 P like S but must be a power of 2
 C single character
 A hexadecimal address
 F string
 N cache level (1 <= N <= 5)
 T cache type (u=unified, i=instruction, d=data)
```

An example Dinero command is this:

> ./dineroIV -l1-isize 16384 -l1-iassoc 4 -l1-ibsize 32 -l1-irepl l -l1-dsize 32768 -l1-dassoc 2 -l1-dbsize 16 -l1-drepl f -l1-dwalloc a -l1-dwback a -informat d < cc1.din > sample.out

Note the re-directions utilized:
{-informat d < cc1.din} This is the input to the command line which is a trace file, this trace file should be available in the same folder.
{cc1.din > sample.out} This is the redirected output to be stored in a file called sample.out

Use your favorite text editor, such as emacs, to see the result:
> emacs sample.out

```
---Dinero IV cache simulator, version 7
---Written by Jan Edler and Mark D. Hill
---Copyright (C) 1997 NEC Research Institute, Inc. and Mark D. Hill.
---All rights reserved.
---Copyright (C) 1985, 1989 Mark D. Hill.  All rights reserved.
---See -copyright option for details
---Summary of options (-help option gives usage information).
-l1-isize 16384
-l1-dsize 32768
-l1-ibsize 32
-l1-dbsize 16
```

```
-l1-isbsize 32
-l1-dsbsize 16
-l1-iassoc 4
-l1-dassoc 2
-l1-irepl l
-l1-drepl f
-l1-ifetch d
-l1-dfetch d
-l1-dwalloc a
-l1-dwback a
-skipcount 0
-flushcount 0
-maxcount 0
-stat-interval 0
-informat d
-on-trigger 0x0
-off-trigger 0x0
---Simulation begins.
---Simulation complete.
l1-icache
 Metrics                  Total          Instrn          Data          Read          Write          Misc
 -----------------        ------         ------         ------        ------        ------         ------
 Demand Fetches           757341         757341              0             0             0             0
  Fraction of total       1.0000         1.0000         0.0000        0.0000        0.0000        0.0000

 Demand Misses             13743          13743              0             0             0             0
  Demand miss rate        0.0181         0.0181         0.0000        0.0000        0.0000        0.0000

 Multi-block refs             0
 Bytes From Memory       439776
 ( / Demand Fetches)     0.5807
 Bytes To Memory              0
 ( / Demand Writes)      0.0000
 Total Bytes r/w Mem     439776
 ( / Demand Fetches)     0.5807
l1-dcache
 Metrics                  Total          Instrn          Data          Read          Write          Misc
 -----------------        ------         ------         ------        ------        ------         ------
 Demand Fetches           242661              0         242661        159631         83030             0
  Fraction of total       1.0000         0.0000         1.0000        0.6578        0.3422        0.0000

 Demand Misses              4248              0           4248          2095          2153             0
  Demand miss rate        0.0175         0.0000         0.0175        0.0131        0.0259        0.0000

 Multi-block refs             0
 Bytes From Memory        67968
 ( / Demand Fetches)     0.2801
 Bytes To Memory          40784
 ( / Demand Writes)      0.4912
 Total Bytes r/w Mem     108752
 ( / Demand Fetches)     0.4482

---Execution complete.
```

3. Assignment

Question 1: Replacement policy (to be done for each of the three traces)
Examine an 8K-byte, 2-way associative cache with 16-byte blocks. What are the average miss ratios for LRU, FIFO, and random replacement policies? Examine the replacement policies by increasing the cache size to 16K-byte. **What conclusions can you draw from this experiment?**

Question 2: Unified and split caches (use cc1.din only)
Compare the cache miss ratios of the following two systems:

- A system with a 32K-byte unified cache
- A system with a 16K-byte instruction-only cache and a 16K-byte data-only cache.

Assume the caches are 4-way set associative, LRU replacement policy and the block size is 32 bytes. **What conclusions can you draw from this experiment?**

Question 3: Multi-level L2 Cache (use spice.din only)

Calculate the AMAT for 2-level cache, LRU replacement policy, with:

- level-1 cache size of 16KB and level-2 cache size of 64KB
- level-1 cache size of 32KB and level-2 cache size of 128KB

Assume direct-mapped for level-1 and 4-way set-associative for level-2 with block size of 16 bytes. The hit time for level-1 is 1 clock cycle and for level-2 is 8 clock cycles. Let the miss penalty for level-2 be 50 clock cycles. **What can you say about the performance of the two cache organizations?**

4. Deliverables

A report where you explain what you did and the results you got. Be concise and clear; however, do use tables and plots as you deem necessary. Include at the end of your report the output of your simulations for **only one run** of Dinero (any one you do during these experiments).

5. References and credits

[1] This assignment was inspired from the teaching materials of Avinash Kodi of Ohio University.