

Embedded Systems

Laboratory 3 – Cache Memories¹

1 Introduction

1.1 Goals

After this laboratory exercise, you should understand the basic principles of cache memories, and how the different parameters of a cache memory affect the efficiency of a computer system.

1.2 Literature

Chapter 3.5 in Wolf: *Computers as Components*.

2 Preparations

Read the literature and this laboratory exercise in detail, and solve the home assignments. Note that you must solve the home assignments, or you will not be allowed to start the laboratory exercise.

2.1 Home Assignment 1

The following C program contains two subroutines which return the sum of all the matrix cells. The only difference between the two subroutines is that they visit the matrix elements in a different order. This may seem unimportant, but with a cache memory, it may make a big difference.

```
#include <stdio.h>
#define N 12

int A[N][N];

int *magic_address = (int *) 0x7fffffff0;

/* This function clears the contents of the D-cache and resets
   statistics */
void flush_cache(){
    *magic_address = 1;    /* Flush D-cache */
    *magic_address = 2;    /* Reset Cache Statistics */
}

/* This functions dumps cache statistics to file */
void dump_cache_stats(int tag){
    *magic_address = 3 | (tag<< 16); /* Dump cache statistics with tag*/
}
```

¹ Authors: Mats Brorsson and Jan Eric Larsson (edited by Ingo Sander for the 2004/2005 version of the course)

```

/* Initialize the matrix */
void InitMatrix (int Matrix[N] [N]) {
    int i, j;

    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            Matrix[i] [j] = i+j;
        }
    }
}

int SumByColRow (int Matrix[N] [N])
{
    int i, j, Sum = 0, Time;

    flush_cache();
    for (j = 0; j < N; j ++ ) {
        for (i = 0; i < N; i ++ ) {
            Sum += Matrix[i] [j];
        }
    }
    dump_cache_stats(1);
    return Sum;
}

int SumByRowCol (int Matrix[N] [N])
{
    int i, j, Sum = 0, Time;

    flush_cache();
    for (i = 0; i < N; i ++ ) {
        for (j = 0; j < N; j ++ ) {
            Sum += Matrix[i] [j];
        }
    }
    dump_cache_stats(2);
    return Sum;
}

main ()
{
    int a, b;

    InitMatrix(A);
    a = SumByColRow (A);
    b = SumByRowCol (A);
    printf ("The sums are %d and %d\n", a, b);
}

```

Study the code carefully so that you understand in what order the matrix elements are used. Also explain in what order the matrix elements are located in physical memory. The source code can be found on the course webpage.

3 ARM simulator with a Harvard cache model

The ARM simulator (ARMuLator) used in the course has been augmented with a Harvard cache model, i.e., a model with separate instruction and data caches. It is a (more or less) clock cycle true model which means that cache miss penalty cycles are accounted for. It models a write back, allocate-on-write cache with configurable number of sets and address mapping.

3.1 Cache configuration

The default configuration is shown in Table 1 below.

Table 1: Default configuration values for the cache model.

| Parameter | Default value |
|-----------------------------------|---------------|
| Use victim cache | False |
| Use data cache | True |
| Number of instruction cache lines | 128 |
| Instruction cache line size | 8 |
| Instruction cache set size | 1 |
| Number of data cache lines | 128 |
| Data cache line size | 8 |
| Data cache set size | 1 |
| Victim cache size | 4 |

A cache according to the configuration above (the instruction and data caches are in this example equally configured) is thus 1024 bytes large and direct mapped since there is only one block in each set. A 2-way set associative cache would have a set size of 2. A fully associative cache would have the same set size as the total number of lines. In the cache of a set-associative address mapping, a random replacement policy is used.

The model also includes the possibility to use a victim cache, which is a small buffer used to hold recently evicted cache blocks, but this is out of the scope for the course and it will therefore not be used.

The timing model takes into account the time it takes to fetch blocks from the main memory. A large cache block takes a longer time to fetch than a small. It does not, however, take cache lookup timing into account. A large cache, and particularly if a large associativity is used, will have a longer lookup time and this is not reflected in the timing model.

The cache can be configured by putting a file named `cache.cfg` in the same directory as the project for the program that is to be run on the simulator. Here is an example of how a configuration file might look like:

```
USE_VICTIM_CACHE 0
USE_DCACHE       1
ICACHELINES      128
ICLINESIZE       8
ICSETSIZE        1
DCACHELINES      128
DCLINESIZE       8
DCSETSIZE        1
VICTIMSIZE       4
```

```
#####
Do not modify anything but numerical values above this line!
```

Explanations of configuration parameters. The values below are the default when this file is not present.

```
USE_VICTIM_CACHE 0      1 if victim cache is used, 0 otherwise
USE_DCACHE       1      1 if data cache is used, 0 otherwise
ICACHELINES      128    Number of I-cache lines
ICLINESIZE       8      Line size in bytes of I-cache
ICSETSIZE        1      Numbers of blocks in set:
                       1 = direct mapped
                       2 = 2-way set associative
                       ICACHELINES = fully associative

DCACHELINES      128    Number of D-cache lines
DCLINESIZE       8      Line size in bytes of D-cache
DCSETSIZE        1      Numbers of blocks in set:
                       1 = direct mapped
                       2 = 2-way set associative
                       ICACHELINES = fully associative

VICTIMSIZE       4      Numbers of blocks in victim cache
```

It is the first nine lines that are important for the configuration. The rest of the lines are only used for explanatory purposes.

3.2 Using the cache model

You need to modify a configuration file for the ARMulator in order to use the cache model. Copy the file `armul.cnf` that can be found in the same directory as the executable files for the ARM project manager and debugger. This is probably in `C:\WINAPP\ARM251\bin` on the lab computers. Look carefully, it might not be that the extension `.cnf` is visible in Windows.

Copy the file `armul.cnf` to the directory where your homework program is. Edit the file and find the line which says `HarvardCacheInit`. Remove the two semicolons in front of it. Add two semicolons in front of the line `PeriferalInit`. If you can find neither of these, add this word on a separate line after `EarlyModels`.

The cache model can be controlled a little from the program run on the simulator through magic address which is 0x7ffffff0. Table 2 shows the different values which can be written to the magic address in order to control the cache model. Any other value will be ignored.

Table 2: Controlling the cache model from the user program.

| Value (C syntax) | Meaning |
|------------------|--|
| 0 | Flush the contents of the instruction cache |
| 1 | Flush the contents of the data cache |
| 2 | Reset cache statistics |
| 3 + (x << 16) | Dump cache statistics with tag x. |
| 4 | Dump statistics and switch back to no cache simulator. |

3.3 Cache model statistics

The model collects statistics about processor cycles and cache misses and references. When the debugger exits, a file called cache_stats.txt is created. The following is an example of how it might look.

```
Current date and time is: Wed Jun 21 15:38:42 2000
```

```
Simulation of ARM cache system
*****
```

```
Clock frequency and memory data
=====
```

```
Clock frequency (MHz): 100
Clock period in nanoseconds: 10
See memory map for information on memory latencies
```

```
Execution statistics
=====
```

```
Number of elapsed clock cycles: 301238
Elapsed time in microseconds: 3012
```

```
Instruction cache parameters
=====
```

```
Size: 1024 bytes
Line size: 8 bytes
Number of sets: 128
Lines in a set: 1
```

```
Instruction references: 201636
Instruction cache misses: 3198
```

Instruction cache hit ratio: 0.984140

Data cache parameters

=====

Size: 1024 bytes

Line size: 8 bytes

Number of sets: 128

Lines in a set: 1

Data references: 18474

Data cache hits: 641

Data cache hit ratio: 0.965303

Victim cache references: 641

Victim cache hits: 2

Victim cache hit ratio: 0.003120

4 Function of a Cache Memory

A cache memory is a memory that is smaller but faster than the main memory. Due to the locality of memory references, the use of a cache memory can give the effect on the computer system that the apparent speed of the memory is that of the cache memory, while the size is that of the main memory.

4.1 Assignment 1

Create a project, and build the C program using the code of Home Assignment 1. Then copy the configuration file `armul.cnf` to and modify it to enable the cache model and disable the I/O model used in lab 2 as described above. Create a `cache.cfg` file and add configurations to model an instruction cache which is 32 bytes with 4 bytes line size and a data cache which is 256 bytes but with 16 bytes line size. Both caches should be direct mapped.

Then start the ARM debugger with the program loaded. You now need to configure the debugger to use the original `armulator.dll` instead of `thaimulator` which was used in lab 2.

Verify with the disassembly function in the debugger that the code performs what you think it should do. Note the differences between the two versions of matrix addition.

Run the program, verify that it works and then exit the debugger. Not until you exit will the cache statistics file be closed.

4.2 Assignment 2

Study the cache statistics from the execution in assignment 1 and in particular the instruction cache statistics. Also look at the disassembly view. Answer then the following questions:

- How many distinct instructions are there between the call to flush the cache cache and the call to dump statistics?

- How many instructions are executed?
- Explain the differences for the two versions of matrix element additions.
- Is the number of cache misses what you would have expected? Why? / Why not?

4.3 Assignment 3

The parameters of the cache memory can be changed to test the effects of different cases. Investigate the effects of different parameter settings.

- Explain the following: cache size, block size, number of sets, write policy, and replacement policy.
- If a cache is large enough that all the code within a loop fits in the cache, how many cache misses will there be during the execution of the loop? Is this good or bad?
- What should the code look like that would benefit the most from a large block size?

5 Cache Efficiency

The actual efficiency gained by using a cache memory varies depending on cache size, block size, and other cache parameters, but it also depends on the program and data.

5.1 Assignment 4

Compile the C program of Home Assignment 1 with the Debug configuration. Execute and investigate a data cache size of 128 bytes, block size 16, and 1 line per set. Study the subroutines SumByColRow and SumByRowCol. Explain carefully in what order the memory addresses are visited by the two subroutines.

- Execute the program and study how many cache hits the two subroutines have. Is there a difference? Why?

5.2 Assignment 5: Competition!

Find the most effective organizations for an instruction cache of 256 bytes and a data cache of 256 bytes for the program in the home assignment in which only the most efficient matrix element function is used. Remove all flush cache and dump statistics so that you measure on the entire program.

Hand in your answer to the lab assistant with your name(s) on it. The three best solutions will be honorary mentioned.

6 Conclusions

Before you pass the laboratory exercise, think about the questions below and explain to your supervisor:

- What is the general idea with cache memory?
- What is a block (line)?

- How does block size affect the efficiency of a cache?
- How fast is a cache memory? How fast is a DRAM?
- Do the optimal cache parameters depend on the program code?
- How can one select good cache parameters?
- Is it possible to change cache size on a PC? On a Mac?