Report on

Cache Simulator

Rahul Mittal (2012CSB1027)

Kaushal Yagnik (2012CSB1039)

Gaurav Mittal (2012CSB1013)

Part 1

Cache Simulator in C

The Simulator takes instructions from the supplied input file and simulates the performance of a Cache and displays various results likes Hits, Misses, etc.

The Cache implemented is Level 1 Data Cache and it runs on parameters supplied while running the Simulator.

The cache simulator works as following:

The instruction from the file is read. The instruction is decoded and depending whether it is read or write; the data is moved in the memory. The cache exists between the processor and main memory. The cache has several cache lines (blocks) and each cache line has many slots for storing data.

Each block is represented by an object of “myblock”. Each address in myblock is stored in an array “myaddr”.

Let us assume the data is to be accessed has address SRC.

The cache line to which it is mapped is determined by following formula:

(Cache address) c_address = (SRC / block size) % associativity

The data is then scanned in c_address to c_address + associativity.

Depending on the instruction, operation read/write is performed. Corresponding events (Hit/Miss, Dirty Block, etc.) are noted.

If data is not present in cache, the block selected by replacement policy is flushed with required data.

After simulation ends, Output is printed.

To run the simulator:

```
$ ./a.out -assoc <associativity> -block <blocksize_in_bytes> -cap <capacity_in_kbytes> -repl <replacement_policy_LRU/random> < <input_file> > <output_file>
```

Part 2

Cache Simulator integration in Sim-Fast

Sim-Fast is an open source low level cpu simulator in simplescalar package that has no cache simulation but higher MIPS.

Every component of a computer such as memory, cache, opcode-decode, etc are present in different source files.

Different simulators that use these components just “include” them in their source.

We read the code of sim-fast.

The parameters such as word_t, mem_t, etc were defined using typedef.

For example, a 32 bit word size word_t would be int, for 8 bit it would be char.

We found all of that by searching all the source files using grep command from the terminal in the simplesim-3.0 folder.

The functions read and write were pointed to mem_read and mem_write using typedef command.

We tried to implement our cache simulator at this place.

We could redirect read and write to cache_read and cache_write of our simulator and our functions can further call mem_read and mem_write.

But the problem was that we need to flush the whole block of cache in case of a miss. We need to retrieve data from the memory and in order to do it we need to modify the code of memory.

Memory was in 2 parts, memory.c and memory.h. The modification in header file is new to us so we needed a lot of time.

We tried our best to find some way to implement our cache into it. We spent 48 hours (in 4-5 days) just trying to understand the code of sim-fast.

We made best possible efforts.