RUTGERS

CS211 Computer Architecture

Fall 2020

Recitation 12

Today

- HW6

HW6 – Cache Simulator

- Simulate a L1 write-through cache given input parameters that determine design and replacement policies
- Your program will read trace files
- Given the layout of the cache, you will read each line in the file and determine whether there is a cache **hit** or **miss**

HW6 – Cache Simulator

- Your file should be called `first.c`
- How the program will be run as

```
./first.c <cache size> <assoc> <replacement policy> <block size> <trace file> *
```

-always check if you have enough

* Always check if you have enough arguments and return `error` if not (this case argc == 6)

HW6 – Cache Simulator

`./first.c` **`<cache size>`** `<assoc> <replacement policy> <block size> <trace file>`

- Cache size = number of bytes in cache
  - Check to see if it is a power of 2; return `error` if not

HW6 – Cache Simulator

`./first.c <cache size> <assoc> <replacement policy> <block size> <trace file>`

- Associativity – layout of cache
  - options are:
    - `direct` – direct mapped
    - `assoc` – fully associative
    - `assoc:n` – n-way associative
  - else, return `error`

HW6 – Cache Simulator

`./first.c <cache size> **<assoc>** <replacement policy> <block size> <trace file>`

- Given associativity, you should be able to calculate
  - Number of sets
  - Associativity (how many cache lines per set)

Associativity	**Number of Sets**	**lines per set**
Direct mapped	cache size/block size	1
Fully associative	1	cache size/block size
N-way	cache size/(n*block size)	n

HW6 – Cache Simulator

`./first.c <cache size> <assoc> **<replacement policy>** <block size> <trace file>`

- Replacement policy – when a set is full and we want to add a new line, which cache line to we replace?
  - Options are
    - `lru` – least recently used
    - `fifo` – first in, first out
  - else, return `error`

HW6 – Cache Simulator

`./first.c <cache size> <assoc> **<replacement policy>** <block size> <trace file>`

- Replacement policy – when a set is full and we want to add a new line, which cache line to we replace?
  - Least recently used – remove the least recently accessed line
    - Must keep track of which lines (determined by tag and set) were accessed last
    - On hit or insertion, that item is now the most recently accessed line
  - First in, first out – remove the first line added to the set
    - Can use “queue” like structure or just maintain order

HW6 – Cache Simulator

`./first.c <cache size> <assoc> <replacement policy>` **`<block size>`** `<trace file>`

- Block size = number of bytes in cache line/block
  - Check to see if it is a power of 2; return `error` if not

HW6 – Cache Simulator

`./first.c <cache size> <assoc> <replacement policy> <block size> <trace file>`

- Trace file – what you will be scanning
  - Format: [illegible]: `<R or W>` `<address 2>`
    - Ignore address 1 – it is the program counter
    - R or W determines if access is read or write
    - Address 2 – the address you will be using to determine the set index and tag
  - The last line of the file is `#eof` to help you determine the end of the file
  - These files are relatively large so runtime won't be instantaneous (but shouldn't be slow either)

HW6 – Cache Simulator

`./first.c <cache size> <assoc> <replacement policy> <block size> <trace file>`

- Trace file – what you will be scanning
  - Address 2
    - NOTE: the addresses are written in HEX, and are **48 bits** long
    - Given the address and its size
      - use the correct datatypes to account for 48 bits
      - use the other parameters to determine the number of bits for the index, tag, and offset (though we don't really care about the offset)
      - use bit operators to extract the information

use bit operators

* Do research about C functions that allow you to do conversions (just do your research in general)

HW6 – Cache Simulator

```
./first.c <cache size> <assoc> <replacement policy> <block size> <trace file>
```

- Trace file – what you will be scanning
  - Example:

```
./first.c 128 assoc:4 lru 8 file1.txt
----------------------------------------------------------------------------------------------
0x804ae19: R 0x8caf41a057de
```

    - R -> you are doing a memory read

HW6 – Cache Simulator

`./first.c <cache size> <assoc> <replacement policy> <block size>` **`<trace file>`**

- Trace file – what you will be scanning
  - Example:

```
./first.c 128 assoc:4 lru 8 file1.txt
----------------------------------------------------------------------------------------------------
0x804ae19: R 0x8caf41a057de
```

    - For the address, we have the following sizes (as bits):
      - Offset size = log2(block size)
      - Index size = log2(number of sets)
      - Tag size = 48 – offset size – index size

HW6 – Cache Simulator

`./first.c <cache size> <assoc> <replacement policy> <block size>` **`<trace file>`**

- Trace file – what you will be scanning
  - Example:

```
./first.c 128 assoc:4 lru 8 file1.txt
----------------------------------------------------------------------------------------------------
0x804ae19: R 0x8caf41a057de
```

    - For the address, we have the following sizes (as bits):
      - Offset size = log2(8) = 3
      - Index size = log2(128/(4*8)) = log2(4) = 2 □ we only have 4 sets of 4 lines each
        - » But we need 2 bits to represent 4 sets
      - Tag size = 48 – 3 – 2 = 43

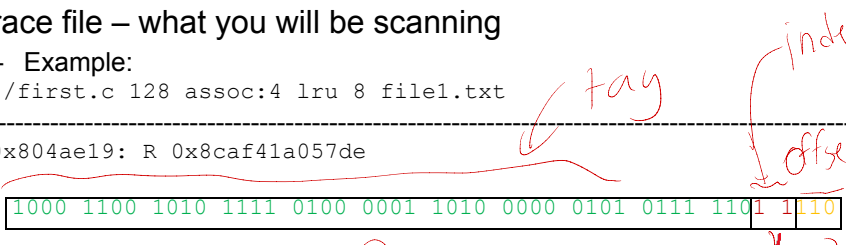
00
01 --> 2 bits to label 4 sets
10
11

HW6 – Cache Simulator

`./first.c <cache size> <assoc> <replacement policy> <block size>` **`<trace file>`**

- Trace file – what you will be scanning
  - Example:

```
./first.c 128 assoc:4 lru 8 file1.txt
-------------------------------------------------------------------------------
0x804ae19: R 0x8caf41a057de
```



Set we will be looking at: 3

HW6 – Cache Simulator

```
./first.c <cache size> <assoc> <replacement policy> <block size> <trace file>
```

- Your job throughout this program is to count and output

```
Memory reads: %d\n
Memory writes: %d\n
Cache hits: %d\n
Cache misses: %d\n
```

- Determine what data structure you will use to design your cache – the replacement policy will really drive your design

array of arrays

HW6 – Cache Simulator

- Hints – when do you increment memory reads, memory writes, cache hits, and cache misses?
  - Don't overthink it – understand what it means to read from memory
  - Regardless of R or W
    - Check to see if the info is in the cache
      - If it is, then you have a cache hit
      - If not, then you have a miss, and you must bring that address into the cache by *reading from memory*
- Use appropriate data types for the values (they should all be unsigned)
  - In our example a tag of 43 bits exceeds the size of normal int values…

HW6 – Cache Simulator

- Please submit your hw6 tar file using
  - `tar xvzf hw6.tar hw6`