COSC 530 – Fall 2021
Programming Assignment 1
Memory Hierarchy Simulator
**DUE: 11:59pm on 10/1**

Your assignment is to write a program that simulates the behavior of a memory hierarchy. The hierarchywill include a data translation look-aside buffer (DTLB), page table (PT), first level data cache (DC), and a second level data cache (L2). The program will read a trace of references from standard input and produce statistics about the trace to standard output. The trace of references has the following format:

<accesstype>:<hexaddress>

where <accesstype> can be the characters:

R - indicates a read access
W - indicates a write access

and <hexaddress> is the starting address of the reference expressed as a hexadecimal number. I have provided an example trace in the assignment starter package (*trace.dat*), which is in the appropriate format.

Before processing the trace file, the memory hierarchy simulator should first determine the characteristics of the memory hierarchy to be simulated. To do so, the simulator will read a configuration file named *trace.config* in the current directory that specifies the configuration for the DTLB, PT, DC and L2. I have provided an example *trace.config* file in the starter package, which indicates the required format (note that the values after the colons are the fields that can change).

Specific requirements are as follows:

- The maximum number of sets that can be specified for the DTLB is 256.
- The maximum number of sets for the DC is 8192.
- The maximum associativity level is 8 for the DTLB, DC, and L2.
- The maximum number of virtual pages is 8192.
- The maximum number of physical pages is 1024.
- The number of sets and line size for the DTLB and DC, the number of virtual pages, and the page size should all be powers of two.
- The maximum reference address length is 32 bits.
- The data line size for the DC should be at least 8.
- The data line size for the L2 should be greater than or equal to that of the DC.
- Blocks that are present in the DC should always be present in L2 (i.e., use an inclusive multilevel caching policy).
- The simulator should use an LRU replacement algorithm for the DTLB, DC, L2, and page table.
- Physical pages should be initially allocated from 0..n-1, where n is the number of physical pages.
- When a page fault occurs, you should invalidate the DTLB entry and the DC and L2 lines that are associated with the page that was replaced.

You should also implement the following options in the configuration file. Note that even though a particular option or level of the memory hierarchy may be disabled, you still have to fill in the information about the disabled option in the configuration file.

- For both the DC and L2, the simulator can use write-allocate, write-back as well as no write-allocate, write through policies for data writes.
- The simulator will disable virtual to physical address translation if the "Virtual addresses" option is disabled.

- The simulator can optionally disable the DTLB.
- The simulator can optionally disable the L2.

The starter package includes my executable for the simulation, which is named *memhier_ref*. Your output should match my format exactly. You will first print:

a. the configuration of the memory hierarchy

Next, you will print:

b. information about each reference

The fields to be printed for each reference are the virtual address, virtual page number, page offset, TLB tag, TLB index, TLB result, PT result, physical page number, DC tag, DC index, DC result, L2 tag, L2 index, and L2 result. The specified format to print these fields for the full memory hierarchy simulation is given in the form of a C printf format specification below:

```
"%08x %6x %4x %6x %3x %4s %4s %4x %6x %3x %4s %6x %3x %4s"
```

Printing this information will also help you debug problems with your simulator. When virtual to physical address translation, the DTLB, or the L2 cache is disabled, then the output fields that are not relevant should be left blank. Next, you will print the:

c. number of hits
d. number of misses
e. hit ratio

for each level of the hierarchy. Finally, you should also indicate the:

f. number of accesses that were reads
g. number of accesses that were writes
h. ratio of accesses that were reads
i. total number of memory references
j. total number of accesses to the page table
k. total number of disk references

To help diagnose problems with your program, you can examine the trace.log file that is generated in your current directory when you execute my solution. This file displays information about the valid lines in the DTLB, DC, and L2 and the resident pages in the page table after each reference is processed.

When you have completed your assignment, you should upload a gzipped tar file (created with `tar cvzf ...`) with your source files to the Canvas course website by the submission date and time at the top of this assignment. Partial credits will be given for incomplete efforts. However, a program that does not compile or run will get 0 points. Point breakdown is below:

- Configuration file parsed and all options set correctly (10)
- Page table simulation (15)
- DTLB simulation (15)
- DC simulation (15)
- L2 simulation (15)
- Multiple write policies simulated correctly (10)
- Page fault, eviction policies (LRU) for DTLB, PT, DC, and L2 (10)
- Output is in the appropriate format (10)

The following pages show an example configuration file, input trace data file, and their corresponding output with our solution simulator.

*trace.config*

```
Data TLB configuration
Number of sets: 2
Set size: 1

Page Table configuration
Number of virtual pages: 64
Number of physical pages: 4
Page size: 256

Data Cache configuration
Number of sets: 4
Set size: 1
Line size: 16
Write through/no write allocate: n

L2 Cache configuration
Number of sets: 16
Set size: 4
Line size: 16
Write through/no write allocate: n

Virtual addresses: y
TLB: y
L2 cache: y
```

*trace.dat*

```
R:c84
R:81c
R:14c
R:c84
R:400
R:148
R:144
R:c80
R:008
```

*output*

```
Data TLB contains 2 sets.
Each set contains 1 entries.
Number of bits used for the index is 1.

Number of virtual pages is 64.
Number of physical pages is 4.
Each page contains 256 bytes.
Number of bits used for the page table index is 6.
Number of bits used for the page offset is 8.

D-cache contains 4 sets.
Each set contains 1 entries.
```

```
Each line is 16 bytes.
The cache uses a write-allocate and write-back policy.
Number of bits used for the index is 2.
Number of bits used for the offset is 4.

L2-cache contains 16 sets.
Each set contains 4 entries.
Each line is 16 bytes.
The cache uses a write-allocate and write-back policy.
Number of bits used for the index is 4.
Number of bits used for the offset is 4.

The addresses read in are virtual addresses.

Virtual  Virt.  Page TLB    TLB TLB  PT   Phys        DC  DC          L2  L2
Address  Page # Off  Tag    Ind Res. Res. Pg # DC Tag Ind Res. L2 Tag Ind Res.
-------- ------ ---- ------ --- ---- ---- ---- ------ --- ---- ------ --- ----
00000c84      c   84      6   0 miss miss    0      2   0 miss      0   8 miss
0000081c      8   1c      4   0 miss miss    1      4   1 miss      1   1 miss
0000014c      1   4c      0   1 miss miss    2      9   0 miss      2   4 miss
00000c84      c   84      6   0 miss hit     0      2   0 miss      0   8 hit
00000400      4    0      2   0 miss miss    3      c   0 miss      3   0 miss
00000148      1   48      0   1 hit          2      9   0 miss      2   4 hit
00000144      1   44      0   1 hit          2      9   0 hit
00000c80      c   80      6   0 miss hit     0      2   0 miss      0   8 hit
00000008      0    8      0   0 miss miss    1      4   0 miss      1   0 miss

Simulation statistics

dtlb hits        : 2
dtlb misses      : 7
dtlb hit ratio   : 0.222222

pt hits          : 2
pt faults        : 5
pt hit ratio     : 0.285714

dc hits          : 1
dc misses        : 8
dc hit ratio     : 0.111111

L2 hits          : 3
L2 misses        : 5
L2 hit ratio     : 0.375000

Total reads      : 9
Total writes     : 0
Ratio of reads   : 1.000000

main memory refs : 5
page table refs  : 7
disk refs        : 5
```

# Programming Assignment 1 Suggestions

I recommend that you accomplish programming assignment 1 in stages. This means that you can implement a portion of the assignment and test it before proceeding to the next portion. The examples we use to test your code will likely also follow these stages. For each step compare your output with the output generated by my executable using the Unix diff command to ensure your output exactly matches mine. Following this process will help to ensure that you at least get partial credit if you cannot complete the entire assignment.

1. Read in the configuration file, calculate the number of index and offset bits for the different portions of the memory hierarchy, and print out the configuration information.

2. Create a copy of the original *trace.config* file. Mark 'n' after "Virtual addresses:" to indicate that the simulator will not perform virtual to physical address translation. Mark 'n' after "TLB:" to indicate that the TLBs will be disabled. Mark 'n' after "L2 cache:" to indicate that the L2 cache will be disabled. Run your simulation with the resulting configuration and the original *trace.dat*. Note that the references in this file should now be treated as physical, rather than virtual, addresses. Print out the physical address, page offset, physical page number, DC tag, and DC index for each reference.

3. Create your own trace data, where all references are reads. Implement the simulation of the data cache for a direct-mapped organization (set size 1).

4. Enhance the data cache simulation to deal with associativity levels that are greater than 1, where all references are still reads.

5. Implement the simulation of the data cache with the write-allocate and write-back policy, where some of the references include writes.

6. Mark 'y' after "L2 cache:" to indicate that the L2 cache is enabled. Implement the simulation of the L2 cache with the write-allocate, write-back policy. Test using the same set size and line size as the DC with both reads and writes in the reference trace.

7. With the L2 cache still enabled, mark 'y' after "Virtual addresses:" in the configuration file to indicate that you will be processing virtual addresses. Mark 'n' after "TLB:" to indicate that the TLB will be disabled. Implement the simulation of the page table.

8. With the L2 cache and virtual addresses still enabled, Mark 'y' after "TLB:" to indicate that the TLB will be enabled. Implement the simulation of the data TLB.

9. Increment counters during the simulation and print out the summary statistics at the end of the simulation.

10. Implement the simulation of the no write-allocate, write-through policy for both the DC and L2 caches.

11. Test the DTLB, DC, and L2 caches with alternative numbers of sets and associativity levels. Test configurations where L2 uses a larger line size than the DC.

12. Test for invalidating lines in the DC and L2 when a page is replaced.