CDA5155/CDA4150 - Spring 2024
Assignment 2
Memory Hierarchy Simulator

Objectives: Learn how virtual addresses are translated to physical addresses using a data translation look-aside buffer (DTLB) and a page table (PT), how data references are stored and accessed in a data cache (DC) and a secondary cache (L2), how items are replaced using a least-recently used (LRU) policy, how data write policies are implemented, and how lines of a DC and/or L2 are evicted when they reside in a block in a larger level of the memory hierarchy that is replaced.

Your assignment is to write a program that simulates the behavior of a memory hierarchy. The hierarchy will include a DTLB, PT, DC, and a 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 file, */home/faculty/whalley/cda5155exec/trace.dat*, 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. This is accomplished by reading a configuration file in the current directory called *trace.config* that specifies the configuration for the DTLB, PT, DC, and L2. I have provided an example configuration file in */home/faculty/whalley/cda5155exec/trace.config* that indicates the required format (note that the values after the colons are the fields that can change). The maximum number of sets that can be specified for the DTLB is 256. The maximum number of sets for the DC and L2 is 8192. The maximum associativity level is 8 for the DTLB and caches. The maximum number of virtual pages is 8192. The maximum number of physical pages is 1024. The number of sets specified for the DTLB and caches, the number of virtual pages, the page size, and line size for the caches should all be powers of two. The data line size for both caches should be at least 8, should not be larger than the page size, and the L2 line size should be at least as large as the DC line size. Your simulator should use an LRU replacement algorithm for the DTLB, caches, and page table. Physical pages should be initially allocated from 0..*n*-1, where *n* is the number of physical pages. The simulator will employ either a no write-allocate and write-through policy or a write-allocate and write-back policy for the DC. The simulator will employ a write-allocate and write-back policy for the L2. When an L2 cache miss occurs, you should invalidate the DC lines that are associated with the L2 line that was replaced. 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. In addition, you should implement the option in the configuration file for turning off virtual to physical address translation by assuming that the trace contains physical instead of virtual addresses. You also should implement the option of disabling the DTLB. Finally, you should implement the option of disabling the L2 cache. Even though a particular level of the memory hierarchy may be disabled, you still have to fill in the information about the disabled level in the configuration file.

You can access my executable for the simulation in */home/faculty/whalley/cda5155exec/memhier.exe*. Your output should match my format exactly. You will first print (a) the configuration of the memory hierarchy and you will next 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/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 (c) the number of hits, (d) the number of misses, and (e) the hit ratio for each level of the memory hierarchy. Finally, you should also indicate the (f) the number of accesses that were reads, (g) the number of

accesses that were writes, (h) the ratio of accesses that were reads, (i) the total number of memory references, (j) the total number of accesses to the page table, and the (k) the total number of disk references.

My executable will also log when dirty lines are written back to the next level of the hierarchy, a DC line is invalidated when an L2 line is replaced, and when a DTLB entry and cache lines are invalidated due to a physical page being replaced. These events are logged to the *trace.log* file in the current directory.

You should upload the source code for your program in Canvas before class starts on February 1. The header comments should identify you, the assignment, and the command you used to compile your program. You should also use readable and consistent indentation and comments throughout the program. Your program should be able to compile and execute on *linprog.cs.fsu.edu*. Below are an example *configuration* file and an example *trace data* file. On the following page is the corresponding *output*.

configuration

```
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: y

L2 Cache configuration
Number of sets: 8
Set size: 2
Line size: 64

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

trace data

```
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 no write-allocate and write-through policy.
Number of bits used for the index is 2.
Number of bits used for the offset is 4.

L2-cache contains 8 sets.
Each set contains 2 entries.
Each line is 64 bytes.
Number of bits used for the index is 3.
Number of bits used for the offset is 6.

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   2 miss
0000081c      8   1c      4   0 miss miss    1      4   1 miss      0   4 miss
0000014c      1   4c      0   1 miss miss    2      9   0 miss      1   1 miss
00000c84      c   84      6   0 miss hit     0      2   0 miss      0   2 hit
00000400      4    0      2   0 miss miss    3      c   0 miss      1   4 miss
00000148      1   48      0   1 hit          2      9   0 miss      1   1 hit
00000144      1   44      0   1 hit          2      9   0 hit
00000c80      c   80      6   0 miss hit     0      2   0 miss      0   2 hit
00000008      0    8      0   0 miss miss    1      4   0 miss      0   4 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
```

Assignment 2 Suggestions

I recommend that you accomplish assignment 2 in the following stages. The test cases will 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.

(1) Copy the *~whalley/cda5155exec/trace.config* file to your directory. Read in this configuration file, calculate the number of index and offset bits for the different portions of the memory hierarchy, and print out the configuration information. Change the information in the configuration file and retest to ensure that the correct configuration information is printed.

(2) Take the following actions within the 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. Take as input the *~whalley/cda5155exec/trace_phys.dat* file. Enhance your program to print out the physical address, page offset, physical page number, DC tag, and DC index for each reference.

(3) Use the same configuration and create your own trace data sets, 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) Mark 'y' after "Write through/no write allocate:" to indicate that this write policy will be used. Implement the simulation of the data cache with this write policy, where some of the references include writes.

(6) Mark 'n' after "Write through/no write allocate:" to indicate that a write-back/write-allocate policy will be used. Implement the simulation of the data cache with this write policy.

(7) Mark 'y' after "L2 cache:" to indicate that the L2 cache is enabled. Implement the simulation of the L2 cache with a write-back/write-allocate policy. Test for invalidating lines in the DC cache when an L2 line that contains the DC line is replaced.

(8) 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.

(9) Mark 'y' after "Virtual addresses:" in the configuration file to indicate that you will be processing virtual addresses. Mark 'y' after "TLB:" to indicate that the TLB will be enabled. Implement the simulation of the data TLB.

(10) Increment counters during the simulation and print out the summary statistics at the end of the simulation.

(11) Test for invalidating a TLB entry and/or lines in the DC and L2 cache when a page containing a TLB entry, a DC line, or an L2 line is replaced. Check the trace.log file in your current directory that is produced by my *~whalley/cda5155exec/memhier.exe* executable to see when these events occur.

(12) Test your solution with larger sets of trace data to see if your output matches mine.

CDA4150 students are not required to implement steps 8, 9, and 11. However, these students will receive extra credit if they do so.