Computer Systems Architecture

Laboratory Exercise FAQ

1. **Do I need to write a loop that converts the hexadecimal memory address character by character or is there a simpler way?**

   You can use a loop construct if you wish, but I strongly recommend that you look at the fscanf () library function.

2. **How can we implement the cache simulator if you don't give us the data values?**

   To simulate the operation of the fully associative cache you only need to know the addresses of the locations in the Main Memory that are being accessed. The data values that are stored in these locations do not influence the caching algorithm.

3. **Can I write the cache simulator in Java or Python?**

   No. The cache simulator must be written using ANSI standard C.

4. **Can I write my own test trace file?**

   Yes. The trace files are ASCII format text files, and so can be created and edited using editors such as Notepad.

5. **Is there a specific development environment that I must use for my C program?**

   No. You are free to use any of the development environments installed on the Barnes Wallis PC Cluster, or a development environment on your own laptop PC. That said; please remember that the cache simulator must be written as a single source file using ANSI standard C, and that your program must be suitable to be run from a terminal window, providing its formatted output to the terminal window.

6. **Should the CSV formatted output include columns titles?**

   No. Column titles should not be included.

7. **Can I add additional columns to the CSV formatted output?**

   If you wish to add additional columns in your CSV formatted output then these must follow the eight columns defined in Section 4 of the laboratory script.

8. **Can I submit an updated C program and output file after the second laboratory session?**

   You will be assessed on the cache simulation program and results you upload to Blackboard before midnight on the day of your second laboratory session. Files uploaded after this time will not be considered.

Last updated: 25/02/2020.

9. **Do we need to include the C functions in Appendix A and Appendix B in our program?**

   No. The implementations of the Bubble Sort and Cross Correlation algorithms are provided so that you can match the memory accesses in the trace files with the accesses to the data arrays in the algorithms. This will help you analyse the cache simulator results.

10. **Do I need to simulate writing to back to main memory on completion of the program?**

    No. You only need to simulate the memory accesses in the trace files.

11. **Should I use command line arguments (argv[ ]) to specify the trace file filenames or cache modes?**

    When run from the command line your program must report the requested memory access counts resulting from the analysis of both specified trace files (bubble sort and cross correlation) for all cache modes. I suggest that the two filenames are specified as string constants in the program, and that command line arguments are not used.

12. **Why do different bubble sort trace files contain different numbers of memory accesses?**

    The bubble sort algorithm is executed using an array of 700 random 16-bit values, and each execution of the algorithm (to produce a trace file) will be sorting a different set of array values. The number of swap operations required by the bubble sort algorithm, and hence the number of memory accesses, may therefore differ between trace files.

13. **If you compare two trace files for execution of the cross correlation algorithm the memory addresses accessed are different – is this correct?**

    Yes it is correct. The arrays in the bubble sort and cross correlation algorithms may be located at different starting addresses in memory.