Computer Systems Architecture 2022/23
Trace File Generation

This document provides an illustrated example of how the memory trace files are produced.

In this example, the implementation of the bubble sort algorithm presented in Appendix B will be used to sort an array of 5 integer values from lowest to highest. The addresses of the array elements in the external memory of the embedded processor are shown below. The start address of the array (0x03A8) is arbitrary.

Address	**Data**
`0x3AC`	`array [4]`
`0x3AB`	`array [3]`
`0x3AA`	`array [2]`
`0x3A9`	`array [1]`
`0x3A8`	`array [0]`

The array values are initialised as follows: array[ ] = {7, 3, 6, 1, 5}.

The execution of the bubble sort algorithm is shown on pages 2 through 5. Each page corresponds to a given value of the bubble sort iteration counter, `j`, as it counts down from 4 to 0. All array accesses are shown for each iteration of the inner `for` loop, with `i` incrementing from 0 to `j` – 1. The read and write addresses corresponding to the array accesses are shown on the right-hand side of the pages.

Appendix A presents the trace file generated for the sort operation.

The trace files to be analysed in the coursework exercise have been generated by sorting an array of 1500 random integer values, and the corresponding memory trace files typically record over 4,000,000 individual read and write accesses.

array[] = 7, 3, 6, 1, 5

Whole array iteration counter i = 4

```
Inner for loop counter j = 0

if (array[0] > array[1])                   Read:  03A8     Read:  03A9

temp = array[0]                            Read:  03A8
array[0] = array[1]                        Read:  03A9     Write: 03A8
array[1] = temp                            Write: 03A9


Inner for loop counter j = 1

if (array[1] > array[2])                   Read:  03A9     Read:  03AA

temp = array[1]                            Read:  03A9
array[1] = array[2]                        Read:  03AA     Write: 03A8
array[2] = temp                            Write: 03AA


Inner for loop counter j = 2

if (array[2] > array[3])                   Read:  03AA     Read:  03AB

temp = array[2]                            Read:  03AA
array[2] = array[3]                        Read:  03AB     Write: 03A8
array[3] = temp                            Write: 03AB


Inner for loop counter j = 3

if (array[3] > array[4])                   Read:  03AB     Read:  03AC

temp = array[3]                            Read:  03AB
array[3] = array[4]                        Read:  03AC     Write: 03A8
array[4] = temp                            Write: 03AC
```

array[] = 3, 6, 1, 5, 7

Whole array iteration counter i = 3

```
Inner for loop counter j = 0

if (array[0] > array[1])                   Read:  03A8    Read:  03A9


Inner for loop counter j = 1

if (array[1] > array[2])                   Read:  03A9    Read:  03AA

temp = array[1]                            Read:  03A9
array[1] = array[2]                        Read:  03AA    Write: 03A8
array[2] = temp                            Write: 03AA


Inner for loop counter j = 2

if (array[2] > array[3])                   Read:  03AA    Read:  03AB

temp = array[2]                            Read:  03AA
array[2] = array[3]                        Read:  03AB    Write: 03A8
array[3] = temp                            Write: 03AB
```

```
array[] = 3, 1, 5, 6, 7

Whole array iteration counter i = 2


Inner for loop counter j = 0

if (array[0] > array[1])                    Read:  03A8     Read:  03A9

temp = array[0]                             Read:  03A8
array[0] = array[1]                         Read:  03A9     Write: 03A8
array[1] = temp                             Write: 03A9


Inner for loop counter j = 1

if (array[1] > array[2])                    Read:  03A9     Read:  03AA
```

```
array[] = 1, 3, 5, 6, 7

Whole array interation counter i = 1


Inner for loop counter j = 0

if (array[0] > array[1])                    Read:  03A8     Read:  03A9
```

```
array[] = 1, 3, 5, 6, 7

Whole array iteration counter i = 0
```

Appendix A – Generated Trace File

```
R 03A8
R 03A9
R 03A8
R 03A9
W 03A8
W 03A9
R 03A9
R 03AA
R 03A9
R 03AA
W 03A9
W 03AA
R 03AA
R 03AB
R 03AA
R 03AB
W 03AA
W 03AB
R 03AB
R 03AC
R 03AB
R 03AC
W 03AB
W 03AC
R 03A8
R 03A9
R 03A9
R 03AA
R 03A9
R 03AA
W 03A9
W 03AA
R 03AA
R 03AB
R 03AA
R 03AB
W 03AA
W 03AB
R 03A8
R 03A9
R 03A8
R 03A9
W 03A8
W 03A9
R 03A9
R 03AA
R 03A8
R 03A9
```

Appendix B – Source code for Bubble Sort Algorithm

```
/*
 * Filename:      BubbleSort.c
 * Author:        Jack Andrews
 * Student ID:    123456789
 * Date:          20 February 2019
 *
 */

void BubbleSort(short int *array, int length) {

      // Temporary variable used for swapping of elements
      short int temp;

      // Loop counters
      int i, j;

      /* Pass over the whole array on the first iteration. On subsequent
       * iterations, ignore the already sorted upper elements, achieved by
       * decrementing i on each iteration of the outer for() loop.
       */
      for (i = length-1; i >= 0; i--) {

            /* The inner for() loop iterates over the remaining array elements,
             * comparing each and swapping if necessary.
             */
            for (j = 0; j < i; j++) {

            /* Compare the value at index j in the array with the value at
             * index j+1. If array[j] > array[j+1], then swap the elements.
             */
                  if (array[j] > array[j+1]) {

                        // Swap the array elements
                        temp = array[j];
                        array[j] = array[j+1];
                        array[j+1] = temp;

                  }

            }
```