## CS 3340 Computer Architecture

## Optional Homework 7: Cache Comparisons

## Worth 100 points

Objective: Compare two sorting algorithms in terms of number of instructions and cache utilization.

Turn in, zipped together: Bubble sort program, Selection sort program, Document containing counts and commentary

This homework will average in with your other homework grades if you choose to do it. If you don’t choose to do this homework, it will not count against you.

Instructions:

* Modify the Bubble sort demonstrated in class (<https://github.com/kjmazidi/CS3340/tree/master/Code%20Samples/Section%203>) to sort 500 data items which are defined in the .data section. Remove any functionality in the program that is not sorting, such as printing the array before and after.
* Use the same data as you use in the Bubble sort but copy your Selection sort from Homework 4 into a program and modify it to run the same data. The two sorts should be in two different programs and should do nothing but sort the 500 integers in memory.
* Using the Instruction Counter tool, fill in the instruction comparison table below.
* Using the Data cache Simulator tool, fill in the cache comparison table below.
* Answer questions 1-5 below.

Grading Rubric:

20 points: Bubble Sort works (check memory to make sure it sorted)

20 points: Selection Sort works (check memory to make sure it sorted)

20 points: Instruction comparison table (below)

20 points: Cache comparison table (below)

20 points: Answer the questions below

Reminder:

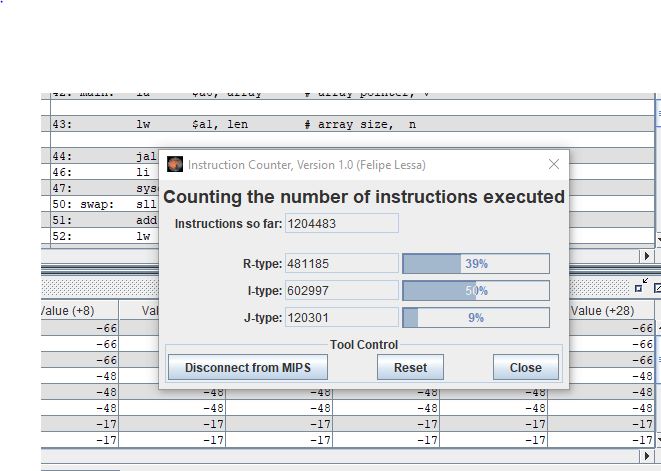
To get to the tools:

* Assemble the program
* Use the Tools menu to find the tool
* Hit the ‘Connect to MIPS’ button on the tool
* Hit the green run arrow at the top of MIPS

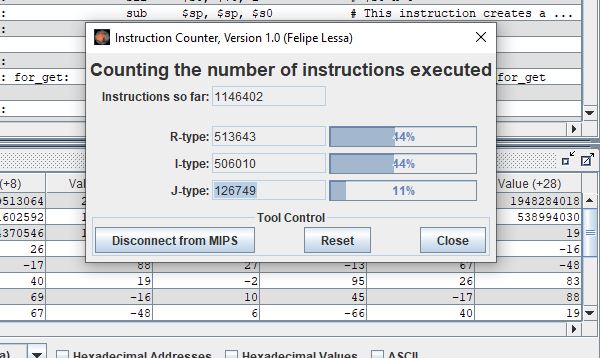
Instruction Comparison Table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Number Instr. | R-type | I-type | J-type | Avg I/item |
| Bubble Sort | 393890 | 157206 | 197378 | 39306 | 2,408‬ |
| Selection Sort | 32251 | 13917 | 14765 | 3569 | ‬2,292 |

Bubble sort:



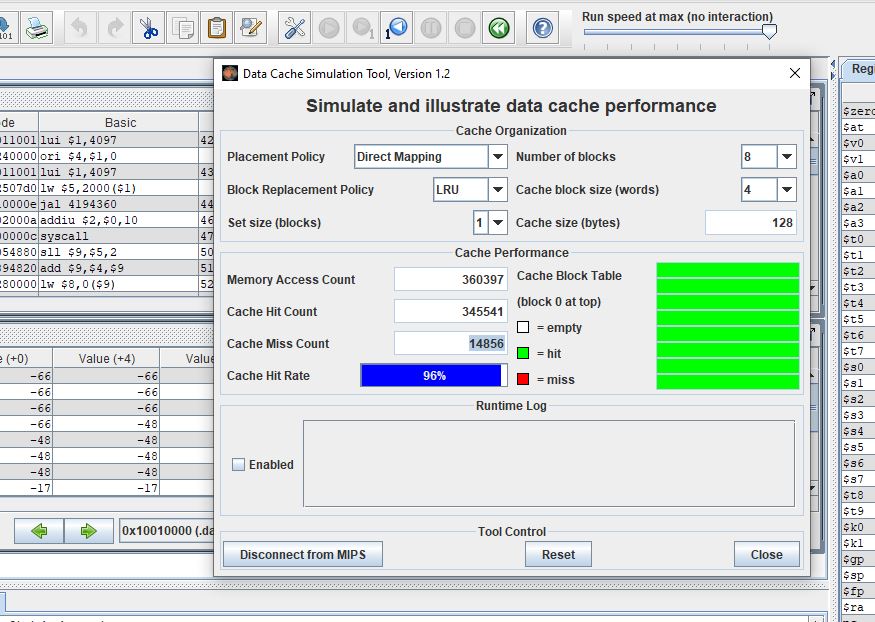
Selection sort:



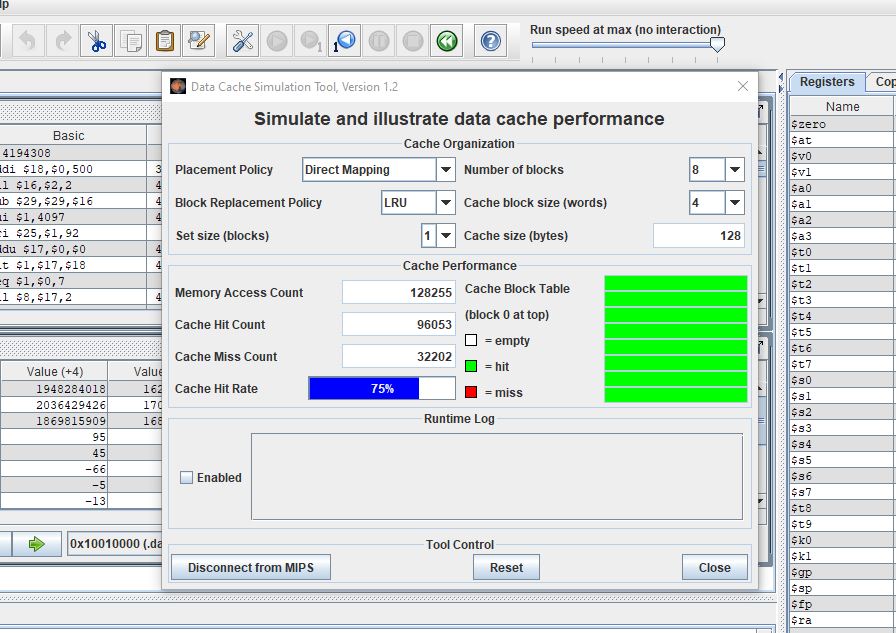
Cache Comparison Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Memory Access Count | Cache Hit Count | Cache Miss Count | Cache Hit Rate |
| Bub Sort | 117624 | 112999 | 4625 | 96% |
| Sel Sort | 4085 | 3055 | 1030 | 75% |

Bubble\_sort:



Selection sort:



Questions:

1. **How similar are the two algorithms in terms of average instructions executed per item sorted? Did this surprise you?**

**Answer:**

The algorithms themselves according to the pseudocode are very similar, so they should be taking the same amount of instructions to execute. The average instructions executed rate for bubble sort is 2,408 and selection sort is 2,292.so that why there is not big difference. This did not surprise me.

1. **How similar is the distribution of R, I, and J instructions for the two algorithms? Comment on why this might be the case.**

**Answer:**

If the R, I, and J instruction distribution is above 50%, the two algorithms are considered similar.

1. **Compare the hit rates of the two algorithms using default settings. Given your understanding of the patterns in which these two algorithms access memory, how do you explain this difference?**

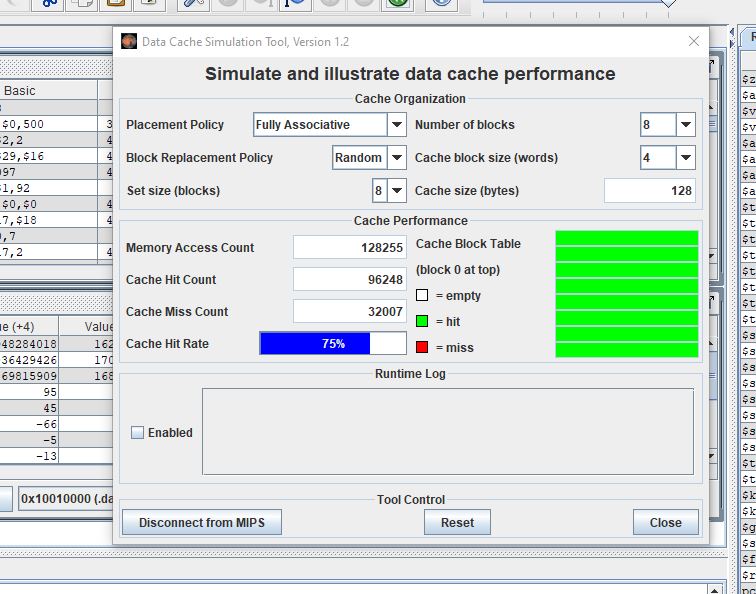
**Answer:**

Yea both algorithms access the cache but there is a difference between the hit rates of the algorithms.

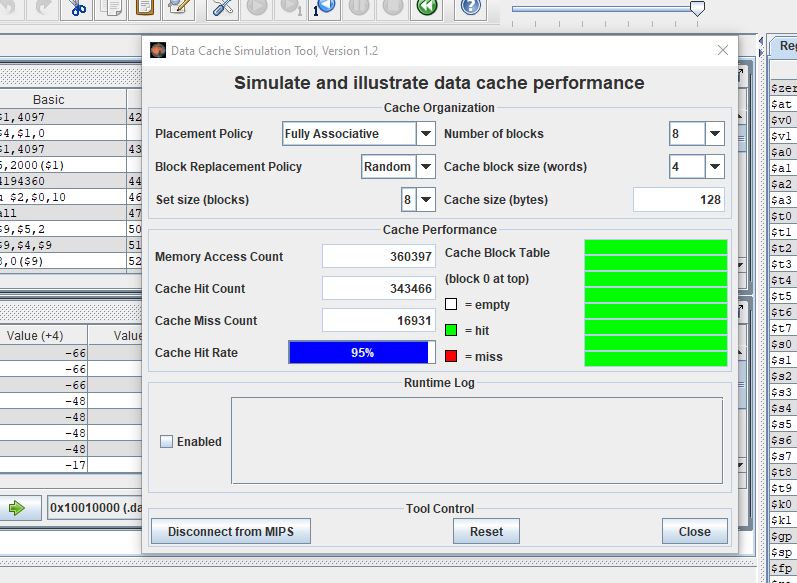
1. Try modifying the placement or replacement schemes? Did you get a different result?

Answer:

Selection sort after modification:



Bubble sort after modification:



Yes I got different results.

1. Try modifying block size or number of blocks. Do not change the total cache size. Did you get different results?

Answer:

Yes. I only got different results after I changed the size of the blocks or number of blocks.