Jovanni Ochoa
Professor Mazidi
CS 2340
Nov 15, 2020

CS 2340 Computer Architecture

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 (last page of this file)

Instructions:

- Modify the Bubble sort, available in the book and my GitHub (https://github.com/kjmazidi/CS_2340/tree/master/Code%20Samples/3-Advanced%20MIPS%20Examples) 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. You can use the data in the bubble sort in the GitHub, and it's just a copy/paste of the first line of data
- Use the same data as you use in the Bubble sort but copy your Selection sort from Homework 3 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	1,204,483	481,185	602,997	120,301	2408.966
Selection Sort	1,043,346	281,358	636,326	125,662	2086.692

Cache Comparison Table:

	Memory Access Count	Cache Hit Count	Cache Miss Count	Cache Hit Rate
Bub Sort	360,397	345,541	14,856	96%
Sel Sort	126,576	94,881	31,694	75%

Questions:

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

The two algorithms were about similar, which I think is pretty good considering I initially thought I was going to have a better selection sort by a long shot. Yes I was somewhat surprised in that they were similar, but only because I expected the bubble sort to be way worse.

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

The distributions weren't too similar. I did have an abundance or I type instructions in both algorithm's, but the amount of R type of instructions differed dramatically. Based on how they were implemented, it's easy to see how I got an extremely different amount of instructions. One thing I can say for sure is that no matter how I made it, I can always see J type instructions being smaller than the rest in both algorithms.

3. 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?

Bubble sort goes one by one comparing each number to every other number and seeing if it is correct and going through the iteration of sorting until all the numbers are sorted correctly. On the other hand, selection sort goes for the first index and then compares that to

all the other numbers in the array, then it goes to the next index and compares all the numbers in the array and it keeps doing this until everything is sorted. Since we are able to see that they have different approaches, we see they access memory in different ways that allow the program to try to guess the next correct path.

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

No, I did not get a different result. Although there is a difference in hit rate, it would be minimal.

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

Yes, I got different results. Both increasing the number of block sizes and the number of blocks dramatically increased the hit rate I got for both of my algorithms; however, it also automatically changed my total cache size which I was told not to do. Unfortunately, I couldn't find how to change my block size or number of blocks without the program automatically changing my total cache size.