Amer Din Homework 7 Cache

Instruction Comparison Table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Number Instr. | R-type | I-type | J-type | Avg I/item |
| Bubble Sort | 1204483 | 481185 | 602997 | 120301 | 2408.966 |
| Selection Sort | 1504266 | 627442 | 628116 | 248708 | 3008.532 |

Cache Comparison Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Memory Access Count | Cache Hit Count | Cache Miss Count | Cache Hit Rate |
| Bub Sort | 360397 | 345541 | 14856 | 96% |
| Sel Sort | 251364 | 196423 | 54941 | 78% |

Questions:

1. How similar are the two algorithms in terms of average instructions executed per item sorted? Did this surprise you?
   1. Between the two algorithms, I found that the average instructions executed per item sorted is not similar. The scale between the average instructions per item sorted is roughly 2000 vs. 3000 and the number of instructions for each is roughly 1200000 vs. 1300000; Therefore, the proportions are not similar, and this also coincides with how the time complexity of bubble sort is o(n), while selection sort is o(n^2).
   2. This did in fact surprise me because I knew there were discreteness between the efficiency of each sort. However, identifying the lower-level constructs definitely show me how selection sort can perform more instructions per sorted item and have a lower cache hit rate.
2. How similar is the distribution of R, I, and J instructions for the two algorithms? Comment on why this might be the case.
   1. The distributions for R, I and J instruction for the two algorithms are quite similar in the fact that the least number of instructions for both is the J-type. The second greatest number of instructions, however between the two algorithms was different from each algorithm. For selection sort, the R-type was much closer to the number of instructions for the I-type, unlike bubble sort. The greatest number of instructions for both algorithms is the I-Type. Selection sort has a greater j-type of instruction mainly because the algorithm requires more jumps in and out of loops, causing an o(n^2) time complexity. The overall consensus on why the distribution is similar might be because the overall number of items is consistent between each algorithm. In addition, the number of instructions read between each is close to each other.
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?
   1. The difference between each cache hit count for each algorithm is quite different; The bubble sort has a greater cache hit rate of 96% vs. 78%. It is important to dissect the differences between the cache hit and mis hit counts for each algorithm. The hit count was 345,541 for the bubble sort compared to 196,423 for the selection sort. But the cache miss count is much greater for the selection sort at 54, 961 compared to 14, 856 for the bubble sort. These 2 inverse relationships signify how the selection sort might have a higher miss count because the swapping takes place in elements that potentially might not even be there, thus causing a lower hit rate.
4. Try modifying the placement or replacement schemes? Did you get a different result?
   1. The way of which I modified the replacement scheme was changing the *Block Replacement Policy* to random as to *LRU* from earlier. I did get a different result once I changed this setting. After changing the setting, in the instance of the bubble sort I noticed an increase of instructions across all types from 1204483 to 1386743. However, the cache comparisons were left unaffected. In the instance of the selection sort, I did not notice any difference in cache comparison or instruction count.
5. Try modifying block size or number of blocks. Do not change the total cache size. Did you get different results?
   1. For this question, I decided to change the number of blocks from 8 to 16. In the instance of the bubble sort, the instruction counter remained unchanged, however in the cache analysis, the hit count increased by roughly 2000 and the miss count also increased by roughly 2000 as well. The overall hit rate remained at 96%. In the instance of the selection sort, the instruction counter data did not change. However, the overall cache hit rate increased from 76% to 84% and the cache miss count increased by almost 15,000. In addition, the cache miss count decreased by almost 15,000 as well. I can holistically conclude that my results were different in both instances.