Prof. Stratos Idreos
CS165: Data systems. Midterm1: 10/11/2017
Duration 2 and 1/2 hours. Return this sheet when done.
Name and HUID:

Guidelines: You may use any printed notes, papers or textbooks you want. No electronic devices are allowed. You need to score 100% to get full points. There are also 25% bonus points. For each part of the midterm give a step by step description and explanation of how you came up with your solution and how the solution works. In addition, explain in detail why you think it is the best choice, what other solutions exist and why you did not opt for them. Give as much information as possible so we can see your thought process. It is OK if your answer is not all the way complete and perfect as long as you clearly articulate the main tradeoffs and design points. Even in cases where you cannot reach an exact solution but you do have a feeling of the kinds of tradeoffs that exist, you should describe those as clearly as possible as this is a big part of the design process and will give you a lot of points (50%). In addition, if after you finish describing a solution you realize that this was actually not the optimal solution, please describe as much as possible why this is the case and what your new intuition says. Such answers will get the full credit if the design considerations that are described in the end are correct and complete. If you make assumptions for issues not covered by the set-up we provide, then you should be clear about what those assumptions are. There are purposely small "gaps" in the design below in some of the questions: you are expected to fill them in and describe the overall design and tradeoffs. For example: "Question X asks for…however, we also need to know the design for Z to solve this one because… I will assume that Z is done as follows… which is a good solution when ….because… And will assume Z is done as follows in all remaining cases because….This leaves me with the following tradeoffs…"

Setting: Assume a database with a single table R. The table has K attributes $\{A_1,A_2,\ldots,A_k\}$ and N tuples. Base data is stored one column-at-a-time as fixed-width and dense array. Each array A_i has a fixed width of A_i.width bytes. No index exists by default. Your hardware has a single CPU with 4 cores and a 4 layer memory hierarchy with the last level being persistent memory. Each memory level Mi has size Mi.size bytes and block size Mi.block bytes (the minimum transfer granularity). The following are true:

a) M4.size > M3.size > M2.size > M1.size, M4.block = 4xM3.block, M3.block = 2xM2.block, M2.block = M1.block
b) Total data size < M4.size, Total data size > M3.size, The size of each attribute column > M2.size

The CPU can read from any memory level directly. When data is not found in M1, it looks in M2, then in M3 and finally in M4. When a block is not found in memory Mi we have one Mi miss. When data is not found in Mi, it is subsequently copied there. To make room for a new block in Mi, an existing block is evicted from Mi (with LRU) and copied in Mi+1 (unless Mi=M3 in which case data exists already in Mi+1=M4 so there is no need to copy it).

c) The CPU can write directly only to M4.
d) The TLB can hold up to 256 M4 memory locations.

Part 1: 65% -> You are expected to use only one CPU core for Part 1.

1) Create a query plan for the following query: select A3+A4 from R where A1<10 or A2>20. Your plan should assume that no indexes are available, use late tuple reconstruction, column-at-a-time processing and bit vectors to hold intermediate results. Define the set of DSL operators you will need (their input/output) and write the plan. (5%)
2) Give the pseudocode for each DSL operator in the plan. (10%)
3) Redo the above two steps using an index on column A1. The index in this case is a fully sorted copy of column A1 (stored as a fixed-width, dense array). If you reuse any operators just say so; do not redefine them. (10%)
4) Construct the costs of a) each operator of the two plans and b) the complete plans. As a metric use M2 misses. Describe these costs both using a textual description (2-3 sentences max per operator) and coming up with an equation in big O notation. (15%)
5) Which of the two plans above is best, when and why? (10%)
6) Using Parts 4 and 5 devise a solution that automatically picks the best plan in any scenario. (15%)

Part 2: 60% -> You are expected to use all 4 CPU cores for Part 2.

Assume we have Q queries arriving at the same time. Queries are of the same form as in Part 1. Your goal is to maximize throughput, i.e., the number of queries we can process per second.

1) Using column-at-a-time processing as in Part 1, describe in pseudocode a shared scan operator, shared fetch operator, and shared addition operator that minimizes M2 misses without thrashing TLB. (15%)
2) Do the same for the index in the form of a sorted column from Part 1.3). (10%)
3) Construct again the total costs as in Part 1.4). (10%)
4) Given your shared processing, would you change your answer in Part 1.5? (5%)
5) Assume you can do any kind of design and you also need to respect specific latency requirements for each query as batches of queries arrive with an arbitrary arrival rate: Is your design above still the best one? If yes, explain why. If not, provide a new one (design, plan, costs, pseudocode). (20%)