SST Assignment
VIP – Team Phoenix
Fall 2025
Due Date: October 10, 2025

** If you were unable to compile SST from source, there is a build script in Discord that will compile SST and all its dependencies.

Assignment:

1. Review slides shown in class on 09/24/25, specifically those that go over the various sections of an SST simulation input file.
    a. Feel free to have the demo_1.py script open as you go through the slide deck, because the snippets of the script are from demo_1.py.

2. Run the demo_1.py script for SST.
    a. This is a simple 1 CPU, cache, and memory simulation provided by the SST devs as part of their tutorial program.
    b. Runtime should take <7 µs

3. If you did not add the stats commands to your script, be sure to include them and re-run demo_1.py.

4. Familiarize yourself with the CSV outfile to answer the questions below. It may help to open it in Excel until you feel comfortable parsing CSVs on the command line.

------------------------------------ QUESTIONS ON NEXT PAGE ------------------------------------

Questions:

- Report total read and write requests.
    - Using the ‘req_latency’ row, calculate the average latency. Hint: (sum of latencies / total requests) = 53.31

- What were the total number of cache hits and misses?
    - Report cache hit rate.
        - Cache hit = 2625
        - Cache miss = 375
        - Rate = 87.5%

- What is the maximum number of requests per cycle the CPU is allowed to issue? Hint: look in the py script.
    - Calculate the actual requests per CPU cycle.
        - "max_reqs_cycle" : 2,
        - 0.18

- How many requests were received by the memory?
    - Calculate average latency.
        - 51.93
    - Calculate average outstanding requests.
        - What does the average outstanding request tell us?
        - Average = 3.73
        - The max was 4 so having it at 3.73 means the requests pipeline was nearly full most of the time

“Big Picture” Short Answer

- How did the CPU perform in terms of its requests per cycle vs its maximum number of requests it could have theoretically sent?
    - It did very poorly on the requests per cycle where it was at 9% of capacity. Therefore it had a lot of cycles without requests

- From your calculated cache hit rate, what % of requests leaked to memory? How effective was the cache in reducing memory traffic?
    - Only 12.5% were cache misses which means that we could check the cache instead of memory most of the time saving cycles.

- What do the memory latencies and outstanding requests reveal about the main memory’s role?

    - Memory access takes significantly longer than a cpu cycle so it slows everything down.
- Are you able to identify a system bottleneck?
    - It is related because there was always plenty of requests in the pipeline and the cpu could have done way more requests per cycle than it actually did. Therefore it is a bandwidth issue because it cannot take memory actions quick enough.