Lab Experiment- Python

Objective:

This manual will help you understand and implement core concepts in digital systems design and computer architecture, bridging the gap between high-level algorithm implementation in Python and hardware design in SystemVerilog.

Prerequisites:

- Basic understanding of programming concepts
- Python
- Text editor (e.g., nano, vim, or gedit)

Problems:

Remember the size of datatypes of int in python.

1. Implement 32-bit Booth's Multiplier in python.
2. Implement 32-bit unsigned non-restoring division algorithm in python.

Task:

Implement a simple cache simulator to model the behavior of a direct-mapped cache in Python.

- The cache simulator should model a direct-mapped cache with 128 cache lines.
- The simulator should handle read and write operations and keep track of hits and misses.
- Define the cache structure with fields for valid bits, tags, and data.
- Write functions to simulate read and write operations, including checking for hits and misses.
- Implement a function to reset and initialize the cache.
- Test the simulator with sequences of memory accesses and compare the results with expected cache behavior.
- Track and report the number of hits and misses to verify the simulator's accuracy

```python
class CacheSimulator:
    def __init__(self, cache_size: int, block_size: int):
        # Your implementation here

    def read(self, address: int) -> bool:
        # Your implementation here

    def write(self, address: int):
        # Your implementation here

    def get_stats(self):
        # Your implementation here

# Test Cases
cache_sim = CacheSimulator(cache_size=1024, block_size=64)
cache_sim.read(0x0000)  # Expected output: miss
cache_sim.read(0x0040)  # Expected output: miss
cache_sim.read(0x0000)  # Expected output: hit
cache_sim.write(0x0080) # Expected output: miss
cache_sim.write(0x00C0) # Expected output: miss
cache_sim.read(0x0080)  # Expected output: hit
print(cache_sim.get_stats())  # Expected output: {'hits': 2, 'misses': 4}
```