**Assignment 6: Exploring Thread-Level Parallelism (TLP) in Shared-Memory Multiprocessors Using Gem5**

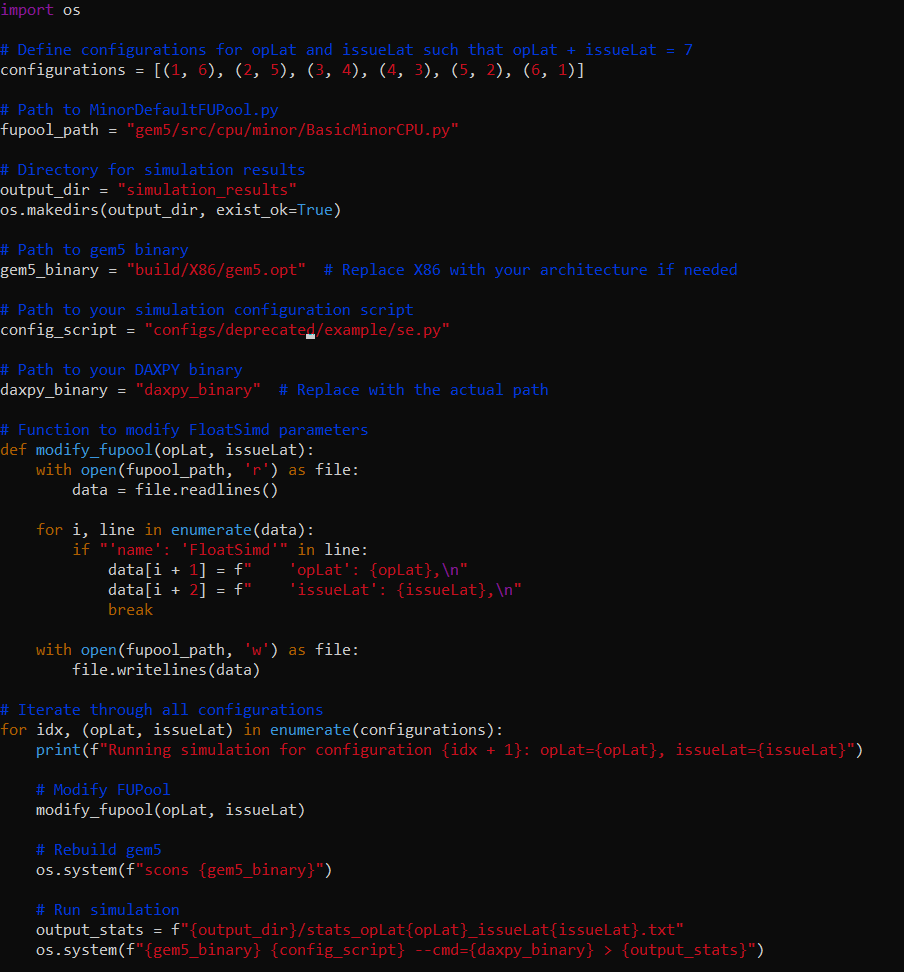
**Part 2: Exploring Shared-Memory Architectures with gem5**

1. **MinorCPU Familiarization**

In the Gem5 directory, the MinorCPU and BasicMinorCPU hold the configuration of the functional units of a CPU. Mostly, in this segment, the basic pipeline stages of a process i.e. fetch, decode, execute and writeback are defined. Additionally, we get to know about opLat and issueLat from these mentioned classes. The opLat is the operation or execution latency of the Functional Unit (FU) which determines the time that blocks the FU for a process. On the other hand, issueLat is defined as the time before the issuing of a new instruction. All functional units have a configuration with combinations of opLat and issueLat and all of them are mentioned within these classes. Various functional units like IntAlu, IntMult, IntDiv, FloatSimd, SimdPredAlu etc. and policies like SingleThreaded, RoundRobin, Random etc. are defined here in these mentioned classes.

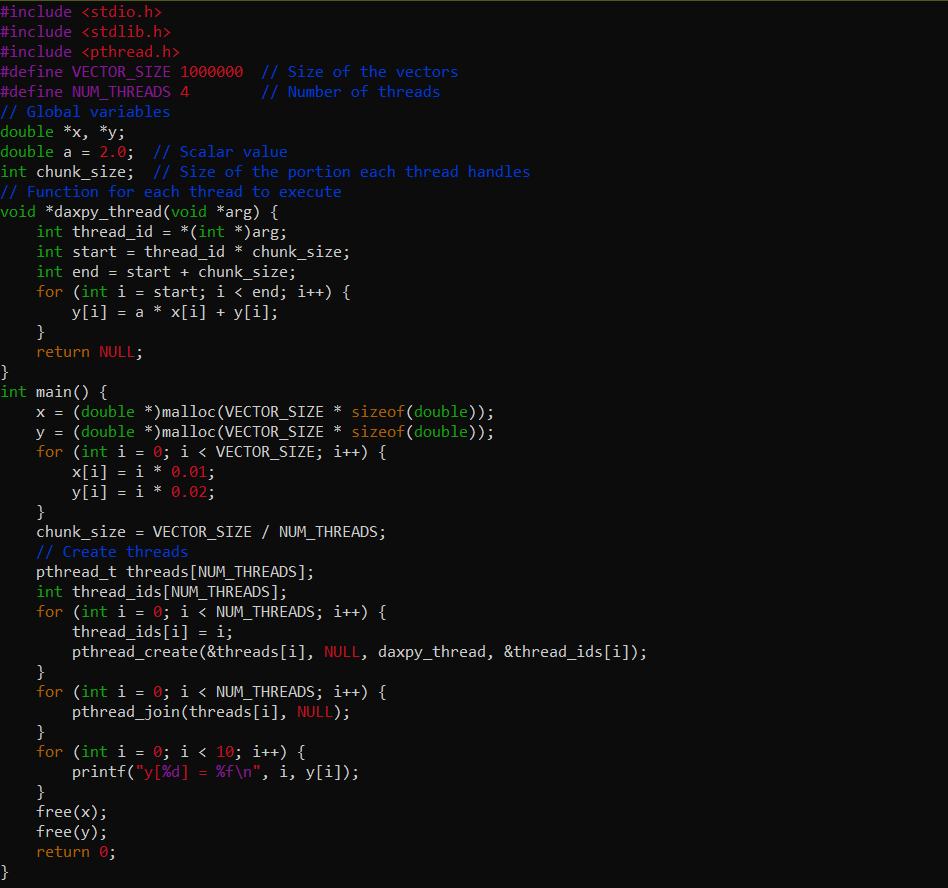
1. **FloatSimdFU Design Space Exploration**

The FloatSimdFU is a functional unit mentioned in the BasicMinorCPU which is capable of handling operations related to floating point and Single Instruction, Multiple Data (SIMD). Here various combinations like (1, 6), (2, 5), (3, 4), (4, 3), (5, 2), (6, 1) are used as opLat and issueLat respectively in this scenario keeping in mind that opLat + issueLat = 7. Here we have automated the testing using a basic Python script by modifying the configuration, rebuilding the Gem5 and running the simulation using the DAXPY binary file to test the efficiency of each configuration. The script is like below.

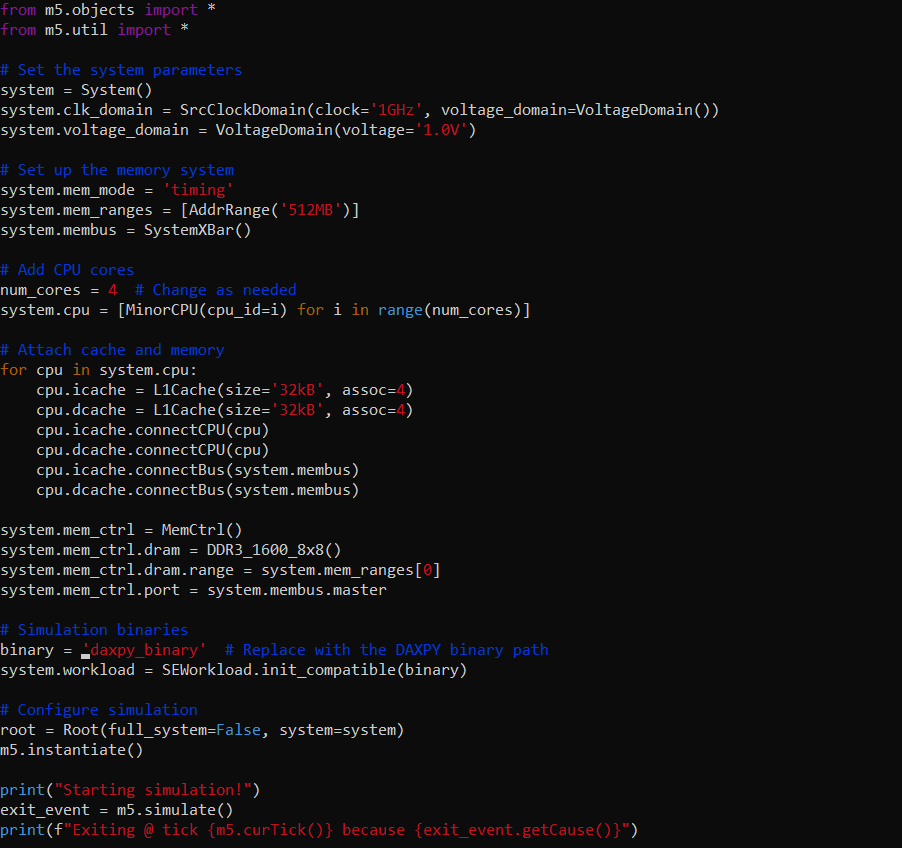


1. **Multi-Threaded DAXPY Kernel Simulation**

DAXPY Kernel has been created in C language where it is mathematically represented as , where and are vectors and is scaler. The created code is like the following:



Once the code has been written it was compiled with GCC compiler to get the binary. Then the created binary is being simulated within Gem5. The simulation script like below.



1. **Performance Analysis**

In the above-mentioned way, the performance of the multi-threaded DAXPY kernel is analysed with different configurations of opLat and issueLat parameters. While examining the overall simulation time, for all of the cases when the opLat increases, the operational latencies also increase which increases the simulation time. On the other hand, with the increase of opLat, the issueLat is decreasing which creates more pipeline stalls and makes less efficient parallel processing.

The increase in parallel speedup can be observed with the number of threads but higher latencies (both opLat and issueLat) decrease the parallel speedup and threads need to wait for the availability of functional units. Similar observations are there for Instructions Per Cycle (IPC) as the increasing values of opLat and issueLat make the idle period longer for functional units. The lower values of Cycles Per Instruction (CPI) are required to make efficient performance as it takes fewer cycles to execute each instruction. But when the latencies like opLat and issueLat increase the instructions may require more cycles for execution of the instructions. Also, the increasing values of opLat and issueLat make resource scheduling inefficient and issues may be observed in thread synchronization. Thus, for effective use of functional units, high FloatSimdFU is required. Higher opLat and issueLat make the synchronization delayed as a result performance of a system is reduced. Synchronization overhead i.e. locks, barriers etc. can reduce speedup in parallel systems.

1. **Comparison and Evaluation**

In Thread-Level Parallelism (TLP), higher latencies (increased values of both opLat and issueLat) reduce the exploitation as more idle time in the functional unit can be observed. Also, the synchronization delays in threads are also increased which actually reduces the speedup using multiple threads. Configurations like opLat=3 and issueLat=4 provide the best performance. On the other hand, with the latencies sometimes the higher number of threads performs well. A higher thread count (8) provides a better performance with moderate latencies whereas a low thread count (2) with lower latencies provides an optimal performance in this scenario.