**ECE550 Post-silicon Validation: Project 2 Part 3 Report**

**-**Abhishek Musku, Sai Teja Gali, Sandeep Goud Abbagouni, Ramakrishna Gopavarapu

**Multiprocessor Instructions and CPU Management:** Implemented atomic XADD and XCHG instructions with randomized LOCK prefixes for multicore synchronization, along with memory fence instructions (MFENCE, SFENCE, LFENCE) for controlling memory ordering. Additionally, implemented CPU binding functionality using sched\_setaffinity() to pin each forked process to specific CPU cores, enabling effective validation of cache coherency and inter-core communication during instruction execution.

**Command Line Arguments**: The program accepts four arguments for test configuration: seed value, number of instructions, number of threads, and optional log filename. The code validates thread count against MAX\_THREADS (4) and handles log file creation with error checking.

**Usage Format**:

./encodeit <seed> <num\_instructions> <num\_threads> [logfile]

**Step 1**: Compiled the code with make, launched it in GDB debugger, and executed it with arguments 12345 25 4 logfile.txt to generate 25 instructions across 4 threads with seed 12345.

run 12345 25 4 logfile.txt

A screenshot of a computer

AI-generated content may be incorrect.

**Step 2**: Set breakpoints at build\_instructions and executeit functions, continued execution

A screenshot of a computer

AI-generated content may be incorrect.

**Step 3**: Hit the executeit breakpoint and used GDB's x/i command to examine and disassemble the generated machine code for thread 0

A computer screen shot of a black screen

AI-generated content may be incorrect.

**Step 4**: Executed the generated instructions step-by-step using GDB's stepi command, observing the actual execution of the instruction sequence including function prologue (ENTER, PUSH operations), data movement (MOV), atomic operations (XCHG, LOCK XADD), and verifying that each generated instruction executed correctly in the processor.

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen with a black screen

AI-generated content may be incorrect.

**Step 5**: Switched to thread 1 execution, where GDB shows breakpoints set for all threads and continues generating instructions for thread 1, demonstrating successful multiprocessor instruction generation with each thread producing its own independent sequence of MOV, XCHG, MFENCE, XADD operations and memory operations with different register assignments and displacements.

A computer screen shot of a black screen

AI-generated content may be incorrect.

**Conclusion**: Successfully implemented a multiprocessor x86-64 instruction validation framework with atomic operations (XADD/XCHG with LOCK prefixes), memory fence instructions, multithread support via fork(), and CPU binding capabilities. GDB debugging sessions validated correct instruction generation and execution across multiple CPU cores, providing an effective foundation for post-silicon validation of cache coherency and inter-core communication protocols.