**Abstract**: The purpose of the lab was to implement a calculator ISA that was able to decode instructions line by line from a text file and execute them. The 6 instructions implemented were add, subtract, load, skip1, skip2 and Print. The final results were successful as the ISA was able to perform all operations from binary code.

**Division of Labor**: We brainstormed and decided to use binary numbers stored in a text file to decode instructions. We were able to parse instructions line by line since the binary code was separated into 8 fields as seen below. We both agreed on code structure and individually wrote code for each function. We later debugged code and modified functions for accuracy

**Detailed strategy:** We separatedevery 2 bits in the binary file line and used them to represent Opcodes, source and target registers and also immediate fields. Our original idea was to read from a MIPS file and translate to binary code but we faced several problems when attempting to parse line by line. The format of our instructions were as follows:

The ISA takes in a text file with 8 bit binary commands line by line. It reads the commands as a string and parses them into opcodes, registers and immediate values. The table below displays this process

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ADD | 0 | 0 | RD | RD | RS1 | RS1 | RS2 | RS2 |
| SUB | 0 | 1 | RD | RD | RS1 | RS1 | RS2 | RS2 |
| LD | 1 | 0 | R | R | NUM | NUM | NUM | NUM |
| SKIP1 | 1 | 1 | 1 | 0 | RS | RS | RD | RD |
| SKIP2 | 1 | 1 | 1 | 1 | RS | RS | RD | RD |
| PRINT | 1 | 1 | 0 | R | R | X | X | X |

**Results:** Add, sub, load, skip1, skip2 and print functions worked correctly. Since we decided on a general structure for the source code for faster debugging, the code was simple, limited to a single cpp file (without classes or header files) and utilized variable assignments, switch cases for each operation and text parsing. Since the data structures utilized were mostly strings encoded as arrays, indices were easy to follow and loops could be easily designed. Minor changes to be included later: max instruction length declarations, overflow checks (signed integers), accepting more registers, hazard detection.

**Conclusion:** I had several projects and exams within the last two weeks so it was a little difficult to meet frequently with Hamza. This resulted in working on the project really late which might have led to undiscovered bugs in the program.

**Appendix I:** Akuma Akuma-Ukpo – 10 hours

Files have been submitted on sakai.