William Ma (Partner: Nicholas Petosa)

Professor Hyesoon Kim

CS 3220: Processor Design

9 March 2017

In terms of workload, Nick and I each did 50% of the workload, with my work being focused primarily on the assembler, later debugging, and then drawing the diagrams that we needed. The assembler ended up being a somewhat large project in and of itself, as I spent a decent chunk of time making it modular in design, extendible, as well as making it highly robust and able to recover from potential errors or typing differences while reading the file. I was able to eventually reproduce the test.mif file after coming up with common parsing methods for various opcodes and classes of instructions. The real problem eventually came up when we tried to link up my assembler and the processor. The fmedian.mif file used a number of instructions and pseudoinstructions not used in the test file, and getting these fixed proved to be a challenge, as we didn't have a .mif file to compare against.

From then on, the problems we ran into were all a result of us being confused about word vs byte addressability for different instructions. We did eventually figure it out, noting that branch and jal instructions were word-addressable, and that alui and mem instructions were byte-addressable. Much of the problem came in the frame originally having inaccurate or incomplete code with regard to this, as well as the assembler not knowing when to convert things to byte addressed offsets or not, especially since parsing methods were shared between instructions. In the end, while the assembler parses instructions word by word, it needs to know when things are referred to by bytes or need to be converted into bytes. This proved the final problem when we did eventually finish debugging, a process which we stepped through together.

To discuss design problems I ran into while working alone, on the assembler beyond the addressability problems, which took up most of our time, what took up the majority of the assembler time was coming up with a functioning system for keeping track of .NAMEs, labels, .WORDs, and after that, a way of consistently keeping track of word addresses and byte addresses and their appropriate conversions at their appropriate times.

Then I went through the process of generating machine code, which was a bit difficult for me, as I forgot much of how to perform bit packing and shifting and masking and whatnot. Eventually though, I remembered what I had done so much in 2110 and 2200, and was able to successfully pack the instructions, with no small help from the fact that the PowerPoint I was looking at's specification for instructions bits turned out to be wrong. After updating, I was able to get that done straight away, and from thereon, the assembler was largely complete. In the end, fmedian takes about 45 seconds, well within requirements.