CDC6600/CDC7600 Simulator

Group Leader: Aaron Easter

Group Members:

Westin Sykes

Aaron Easter

By: Westin Sykes

**Problem Description**

This project included the simulation of the CDC6600 and CDC7600 in either C or C++. This required the design and implementation of the instruction pipeline, scoreboard, functional units, and a logging system which was capable of writing a timing diagram of the code the simulator executes. The first step in implementing the simulator was to design code that could be abstracted to both the CDC6600 and CDC7600. We decided a single class for the functional units could be used. A pipeline was built in to this functional unit class, with the ability to make the segment time equal to the execution time effectively nullifying the pipeline for the CDC6600. This allowed for the functional unit code to be easily abstracted to each system. These functional units had a “clockTick” function which would push instructions down each one’s own pipeline as long as there was no conflict requiring the instruction to not be executed after being issued. The scoreboard object originally was written with abstraction in mind, but it eventually was clear that this was not the best plan, and making two different scoreboards was necessary. The scoreboard for the 6600 and 7600 were very similar with only a few major differences in which functional units it utilized and how it chose which one to use for a given instruction. The instruction pipeline was identical for both the 6600 and 7600. It has the functionality of converting invalid instructions after a long instruction to NO-OPs so the code for the functional units would be slightly simpler. It uses a similar “clockTick” function as the functional units which pushes instructions down the pipe assuming the pipeline had not been stopped due to a conflict. The logging system used to create the timing diagrams, known as “TimingDiagram” in the code, is a public object associated with the simulation. This object stores each issued instruction and the events along its lifetime. These events include issue, start, result, unit ready, fetch and store. It accessed a member variable called “clockTickCount” which stored the number of times the “clockTick” function had been called since the beginning of the simulation to log the time of each event as they occur. Using these main parts along with some other code to pull it all together, these pieces interacted nicely together.

**Results**

**Discussion**