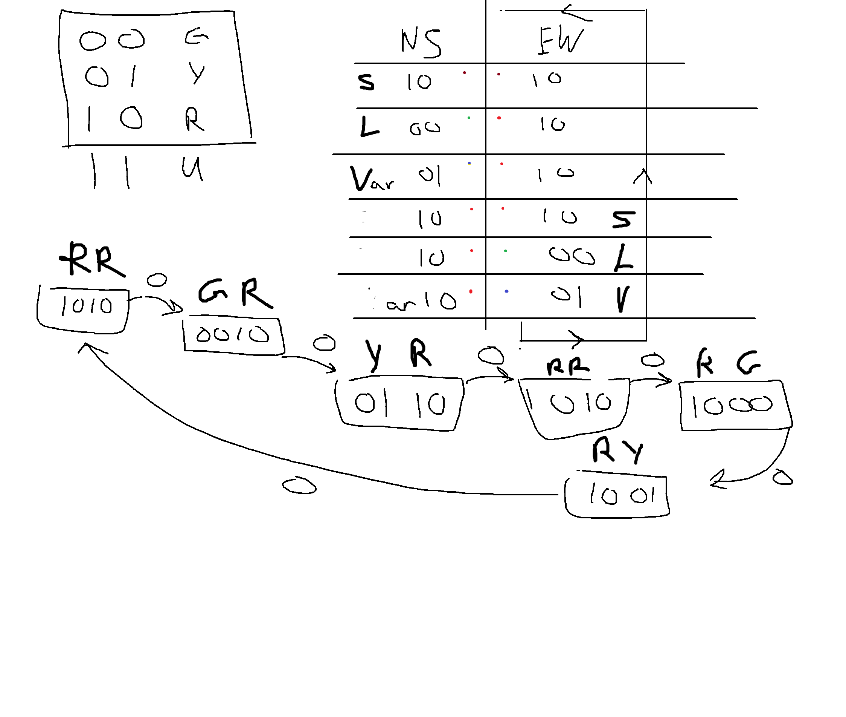
Report – Assembly

Larry S

In this project I faced 3 main issues. First was how to define the problem, properly manage the timings, and allow the user to change the parameters in real time.



Defining how the light sequence should switch on and off was the first step. The whole sequence could be stored in 4 bits to represent all the necessary steps as well as easy rotation using a left shift by 2 that wraps around. Example, 1010 << 2 = 1010, 0010 << 2 = 1000, 0110 << 2 = 1001. As you can see, we can turn the first part of our pattern into the next. This basically covers all the logic of the design. The next issues are also straightforward to address (pun).

In a real-life situation, we would be able to have designated memory addresses for reading clock signals and outside inputs. However, in MARS we need to use the built-in tools that are provided. With MARS I use the syscall 30 which uses java’s time class to count the time since 1970 in milliseconds. Using this we can calculate the next delay by simply adding current time with whatever delay we are currently on. If we are on a red light with a 2000 millisecond delay, and the current time is 1000 millisecond we would wait until the current time is 3000 milliseconds to go onto the next state.

In a similar manner we can deal with user input using the same technique of reading from a designated memory address. However, this does cause issues with the MARS simulator seeing as it breaks whenever you try to use the tool to input data, but it does work perfectly fine to catch interruptions from the keyboard. By checking each loop if the keyboard has a ready signal activated, we can read from the register to get the user input during runtime. This can be used to force the traffic light into different modes such as: emergency, flashing, shutdown, lower delays, etc. But for this application we only check if the user wants to quit ‘q’ or edit ‘e’.