10-2: Met with group (15 min). We decided to either use a memory-to-memory architecture or a stack architecture
10-4: Met with group(1 hour). We decided on a stack architecture and began creating our ISA. We decided on three different types of instructions: A, B, and C. We then decided on which special registers to keep, along with a list of instructions that we need to implement.
10-6: Met with group(2.5 hours). Christian and Will worked on RelPrime code and convention call procedures while I worked on the instruction syntax and semantics so that all three of us would be on the same page. Some decisions I made about the instruction syntax is that

1. All instructions are lower case
2. Most instructions will be named similarly to MIPs instructions
3. The op-codes are organized by type

I also have begun identifying some pseudo-instructions/common operations that will make writing assembly shorter since it is extremely verbose right now
The biggest challenge was deciding on what variations of push to include so that we can load immediates on the stack and get/put values into memory.
10-7: Met with group(3 hours) . Christian and I discussed the ISA that we currently have and push/pop further. We then added ISA instructions and deleted some based on the discussions that we had with you.Will and Christian then discussed/finalized the calling procedures. I created/updated the op-codes for those instructions while Christian updated the RelPrime code to match our syntax. I went through and drew stack for Christian's code to check and see that it made sense. Will typed all completed code and I began translating the finalized code into opcode. We decided on the size of the register stack based on our discussion with you.
10-8: Met with group(30 minutes). We finalized the milestone 1 document and are going to push our stuff. We plan on looking through milestone 2 to make a plan on how to complete it over break. **End of milestone 1.**
10-9: Met with group (1 hour). Discussed feedback and decided what fixes to have done by the time we meet again. We got a rough outline of the memory allocation stack done and changed our reserved memory locations to reserved registers. We decided to add two new instructions, pushR and popR, to our ISA.
10-12: I worked on the project (45 minutes). I added pushR and pop R to our ISA, along with a pushM and popM. I then added some conditionals and loops to our common operations section, but ran into a problem with how to preserve our local variables, which was also brought up in our feedback.
10-13: Met with group(3 hours). We implemented our special sp register into our realprime code, which shortened the code by about a page and a half. We decided to keep the sp register at the top of memory stack and to store our local variables with a modified pushM and popM that adds onto sp with an immediate. Now the programmer can save local variables onto the memory stack and then deallocate the same memory once the function is complete. Christian completed what was left of common operations while Will fixed the realprime code. I began work on Milestone 2 by creating the execution stack component and listing all of the necessary components. I then started writing RTL instructions for all of our instructions. Eventually all three of us worked together to finish single-cycle RTL for all of our instructions except for flip and dupli. We then compiled questions for Sid that we still have for milestone 1 and 2. An important

side note is that we are still considering ways to implement our procedure conventions, and ran into a problem on how to save the return address in the exact order that we specified on the execution stack (return address on the bottom of the stack before the call) . We came up with either an instruction that directly changes the pc counter or using a label that we can push before the arguments. We will finalize that decision after discussing it with you tomorrow.
10-14: Went to office hours to ask questions the group decided on (15 min).Met with group(2 hours). I shared the answers to the questions I'd asked during office hours with Will and Christian. Will and Christian then finished the RTL code for flip, dup, sll, and slr while I started to work on what control signals we would need for our components. Will drew our basic component pictures while Christian and I filled out the rest of the information in component list. Christian and I finalized designing the execution stack component while Will compiled our RTL code into a simplified multi-cycle chart. We decided to finish our summary of how we checked our components tonight and to review/refine our document together tomorrow.
10-15: Met with Christian (1 hour). We fixed several small errors in our required hardware section and then created our process of checking the RTL code. We then turned in Milestone 2.
End of Milestone 2. We began to look at M3's documentation. We decided to begin the work by implementing our components. From our experience, it would probably be best to begin by designing the ALU, register unit, adder, and a mux.This makes sense because our memory, barrel shifter, execution stack, and register file are more complex. A good goal to have would be to have these units tested and implemented in Xilinx by Thursday.
10-16: Met with group (10 min). We divvied up the labs. I get lab 7 memory, Will has lab 6, and Christian has lab 8. We are meeting tomorrow 5-7.
10-18: Worked on memory lab 7 (1 hour).
10-21: Worked on memory lab 7 (3 hours). Completed the memory portion.Met with group(2 hours). Christian worked on making corrections on feedback while Will worked on ALU lab. Will and Christian divided their time creating components while I continued working on the memory lab and completed the control portion. We as a group plan on completing the register file and execution stack together. Worked on memory lab(2 hours). The memory and control components work separately but do not work together. It has to be a problem with how I am connecting them.
10-22: Worked and finished memory lab 7(2 hours). Met with Will(3 hours). I fetched Christian's Xilinx components and went through/checked them in order to see how they worked and to see what in the design document needed changed. I found that the register file currently doesn't have our zero register set to where it cannot be written to. It currently is not always zero. Other than that, everything looked great. Will worked on fixing things from milestone 2. I created our memory component, but am unsure whether or not an address length of 16 will be too much for the FPGA board like lab 7 suggests. If so, we are going to have to adjust our pushM, popM, and program counter accordingly.
10-23: Met with group(2 hours). I demonstrated lab 7 and then completed the memory testbench testing for milestone 3. Christian worked on execution stack while Will began the datapath.We ran into an issue with whether or not popNum was a control or not, but we have decided that it is. Met with Christian(4 hours). I updated most of the components on the list, created the control unit, described all of the control symbols and created the implementation

plan. Will finished ALU testing and updated the component list. Christian finished testing on the execution stack. **End of Milestone 3.** We have decided to perfect our unit testing for the next milestone and to begin implementing our implementation plan after our meeting with you.
10-25: Worked on control list graphs(30 min).
10-27: Met with group (1.5 hours). We worked together on a control signals chart. We are confused about the timing of the execution stack since it is performing more than two operations at a time with our execution stack.
10-28: Met with group(1 hour). We discussed it with you and a major design decision has been made. Instead of popping off two values every time we do an operation, we are performing a peek. Once the operations on the peeked values are performed through the datapath, the value is popped back and onto the stack and other operations occur depending on the instruction. The execution stack deals with putting its final value on like so:

1. The stack always pushes one value onto the stack, what it pushes onto the stack is determined by pushSrc
2. The control signal popAmt determines how many pops to perform on the stack

Christian agreed to change our RTL and execution stack to reflect these changes. I have agreed to finish the multicycle control diagram. Due to tests and special events this week, we have asked for an extension on Milestone 4. I won't be able to work on the project until Wednesday due to seven hours of exams on Tuesday night and Wednesday morning.
10-29: Worked on state transition diagram (3 hours). We had to add additional logic and controls to our datapath in order for beq and bne to work properly. I also changed our RTL to always perform push/popM calculation so that our transition state diagram would not have an empty bubble. I then created as much as I could of our state transition diagram. I plan on finishing that and working on the control unit implementation tonight and having that done before our meeting with you as well as the first sublevel of the implementation plan.