Week 1

1. Discuss what the expected demo of the project with group members
2. Discuss what the project needs to do (general construction of separate part)
3. Search reference of the system on chip and CPU core
   1. Reference Book: Computer Organization and Design RISC-V edition, Digital Design and Computer Architecture, Introduction to Computing System
   2. Reference website or report: ["Longxin" Cup final project](https://github.com/trivialmips/nontrivial-mips), [lowRISC](http://modernhackers.com/build-your-own-risc-v-architecture-on-fpga/) (constructed by University of Cambridge), [Xlinx forum](https://forums.xilinx.com/), [Digilent forum](https://forum.digilentinc.com/), [Design of Digital Circuit](https://safari.ethz.ch/digitaltechnik/doku.php) (course of ETH Zurich)
4. Test the function of Digilent Nexys A7 board
5. Create the [github](https://github.com/dingqy/ELEC222-Project) page and determine the time table of separate parts

Week 2

1. Discuss which instruction set the operating system (xv6 or Linux kernel) needs (RV32IM)
2. Construct the simple Arithmetic Logic Unit Structure on Vivado 2019.1 ([commit detail](https://github.com/dingqy/riscv-core/commit/01af63b6e616400fa81630cb02f4b011370f3002))
3. Update ALU structure and support RV32M extension without verification ([commits on Feb 8, 2020](https://github.com/dingqy/riscv-core/commits/master))
4. Update ALU Control Unit without verification ([commits on Feb 8, 2020](https://github.com/dingqy/riscv-core/commits/master))
5. Construct register file (32 registers following RISC-V specification)

Week 3

1. Construct Control Unit, and test ALU control and register file ([commits on Feb 10, 2020](https://github.com/dingqy/riscv-core/commits/master))
2. Test the ALU with new extension support and fix the bugs ([commits on Feb 9, 2020](https://github.com/dingqy/riscv-core/commits/master))
3. Confirm the function of [SDRAM to DDR Component](https://reference.digilentinc.com/learn/programmable-logic/tutorials/nexys-4-ddr-sram-to-ddr-component/start?_ga=2.78907863.2105935891.1581160391-832371281.1575559289) provided by Digilent on board ([commits on Feb 12, 2020](https://github.com/dingqy/riscv-core/commits/master))
   1. Advantages: It can be directly used to save time
   2. Disadvantages: minimum writing time is 260ns and reading time is 210ns, which causes huge memory operation latency. In addition, it doesn’t have the output for a normal bus that guarantees the communication between memory and CPU core
   3. Problem existing: After reading or writing many times to memory, it will obviously lose data (maybe caused by refresh rate of DDR2 SDRAM in this FPGA)
4. Construct register between each stage used in pipeline processor
5. Test Control Unit and fix the bugs ([commits on Feb 15, 2020](https://github.com/dingqy/riscv-core/commits/master))

Week 4

1. Discuss how the operating system communicate with the CPU (system call)
2. Search information to solve the problem of DDR2 SDRAM, which cause the whole project not to meet the expectation progress.
3. Replace on chip DDR2 RAM with LUT RAM and Block RAM
   1. Can not support the compile file of operating system. Therefore, memory support is necessary

Week 5

1. Integrate all the components to be a pipeline processor with 5 stages ([commits on Feb 27, 2020](https://github.com/dingqy/riscv-core/commits/master))
2. Construct Hazard Unit to solve control dependencies and data dependencies by data forwarding and pipeline stall ([commits on Feb 28, 2020](https://github.com/dingqy/riscv-core/commits/master))
3. Construct Static branch prediction ([commits on Feb 28, 2020](https://github.com/dingqy/riscv-core/commits/master))

Week 6

1. Construct Cache System ([commits on Mar 2, 2020](https://github.com/dingqy/riscv-core/commits/master))
   1. Instruction cache: Single Port Distributed RAM (Xilinx Parameterized Macro)
   2. Data cache Single Port Block RAM (Xilinx Parameterized Macro)
2. Construct communication between cache system and memory by AXI4 full bus ([commits on Mar 3, 2020](https://github.com/dingqy/riscv-core/commits/master))
   1. Reference document: [UG586 MIG 4.2 User Guide](https://www.xilinx.com/support/documentation/ip_documentation/mig_7series/v4_2/ug586_7Series_MIS.pdf), Note 5 of ECE-495/595 in ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT, OAKLAND UNIVERSITY
   2. Tools: Vivado Block Design
   3. IP core: Xilinx Memory Interface Generator, Xilinx AXI4 peripheral, Xilinx AXI4 interconnect
3. Construct communication between cache and cpu core ([commits on Mar 3, 2020](https://github.com/dingqy/riscv-core/commits/master))
   1. Asynchronous FIFO (Xilinx Parameterized Macro)
4. Try to integrate operating system into the SoC
   1. Problem: No GPIO, No USB, Cache system have unknown bugs, boot can not correctly load the system

Further improvement

1. System on Chip
   1. General-purpose input/output (GPIO) to AXI4 interface (memory)
   2. USB driver to AXI4 interface (memory)
   3. PS/2 and VGA driver to AXI4 interface (memory)
   4. Complete verification of all the components
2. CPU Core
   1. Reorder buffer and Out of order execution
   2. Dynamic branch prediction
      1. Reference document: S. McFarling, "Combining Branch Predictors" DEC WRL Technical Report" 1993
   3. Superscalar processor