**Tomasulo Algorithm Report**

Student: Yinzhe Zhang

GTID:#903653372

Email: [zhangyzatsjtu@sjtu.alumni.edu.cn](mailto:zhangyzatsjtu@sjtu.alumni.edu.cn)

# 1.Basic Knowledge of Tomasulo

The Tomasulo algorithm rearranges the order of actual execution of instructions through dynamic scheduling without affecting the correctness of the results to improve time utilization efficiency. It records and detects instructions. Once the operands are ready, they are executed immediately, reducing the possibility of RAW conflicts to a minimum, and eliminating WAR and WAW hazard through register renaming.

Specifically, it can be divided into the following three steps：

**First, Issue.** Take the instruction from the FP Op Queue. If the reserved station is idle (no structure risk), the control mechanism transmits the instruction & sends the operand (rename the register).

**Second, Execution.** If both operands are ready, execute; if not, observe the public data bus and wait for the desired result

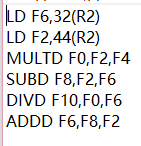
**Third, Write result.** Write the result to all waiting parts through the common data bus.

# 2.Code Structure

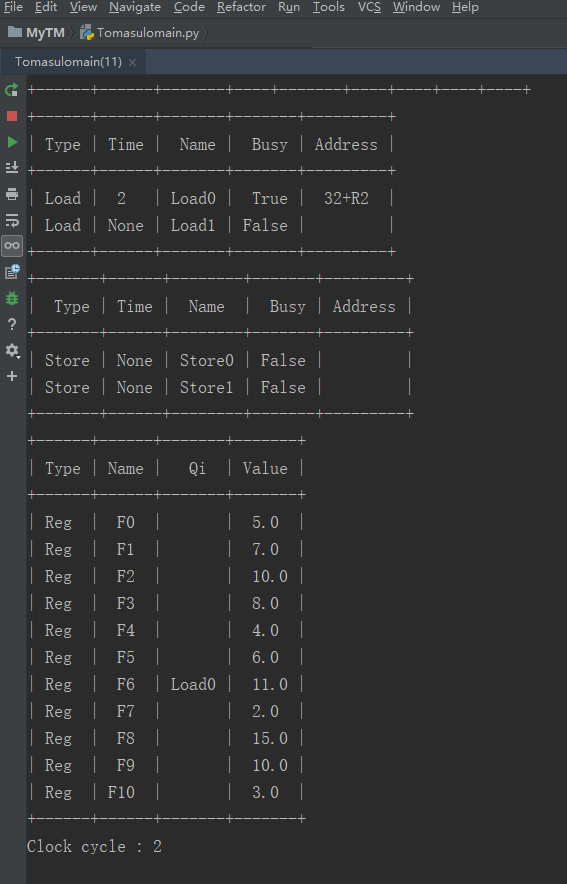
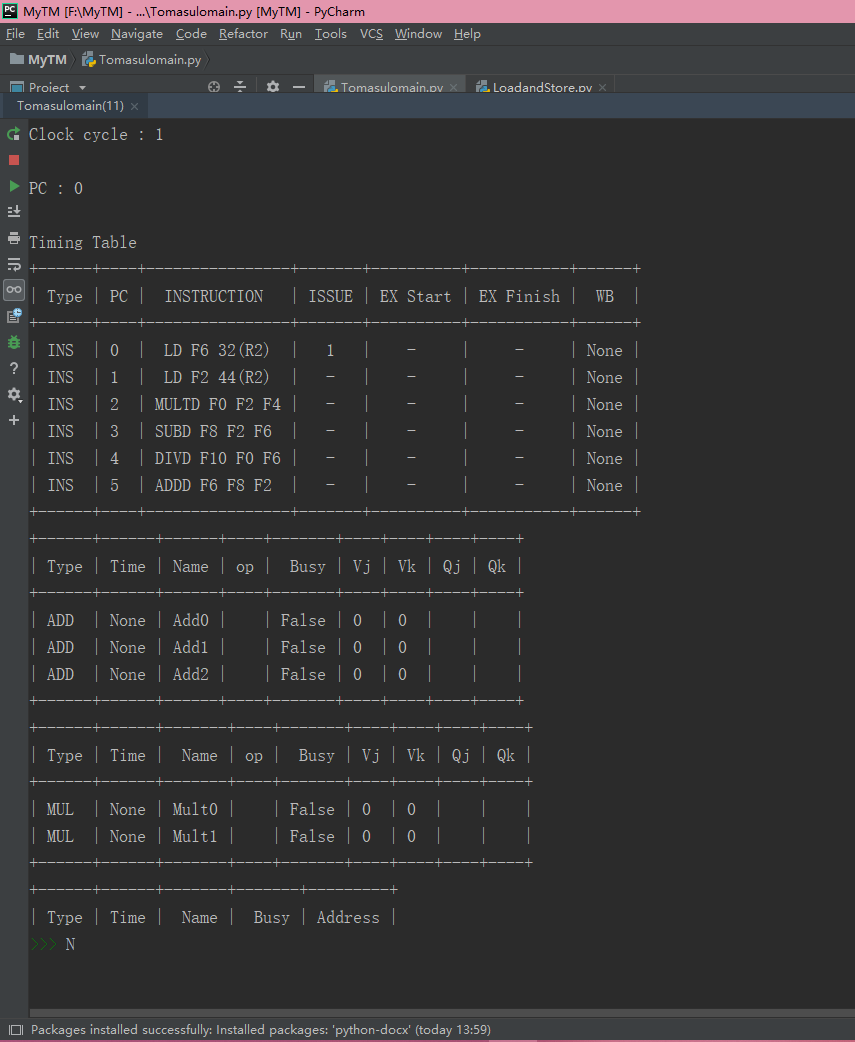
Based on the structural requirements of the algorithm itself, the following schemes can be designed: using classes to implement reservation stations and instruction set queues. Finally, the whole algorithm is regarded as a module similar to a state machine, and the state is updated and output at each step.

# 3.experiment design & result

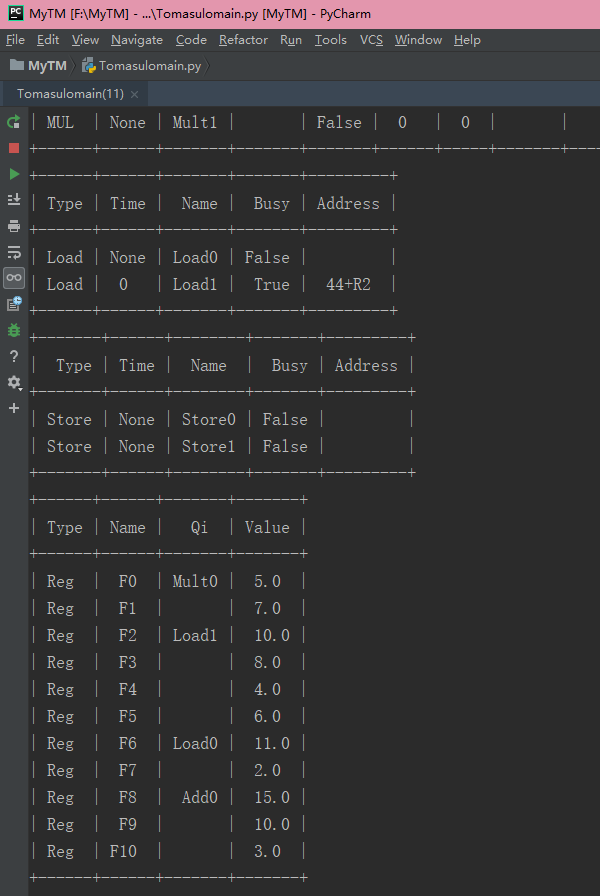
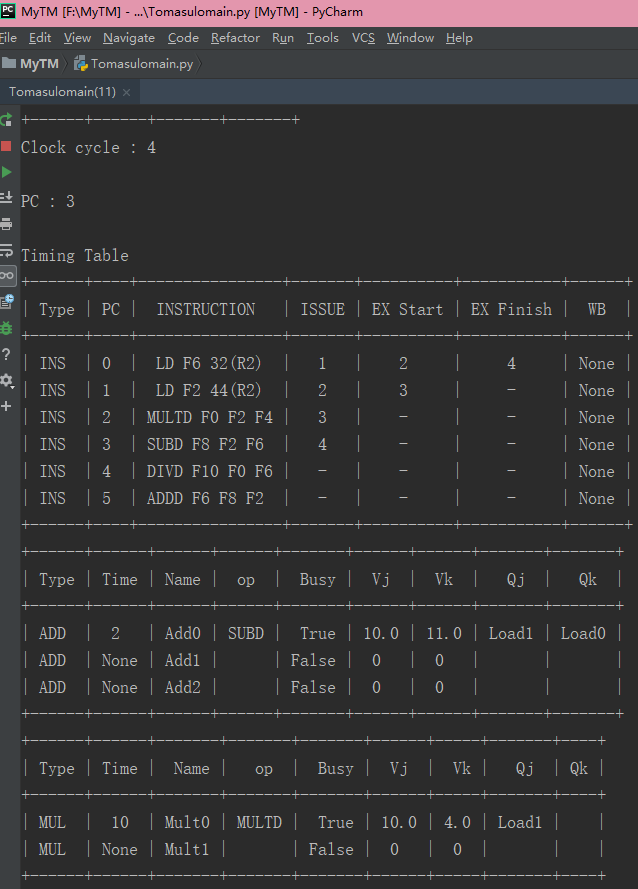
I tested several sequences to see if the algorithm is working well. The first is a long example :

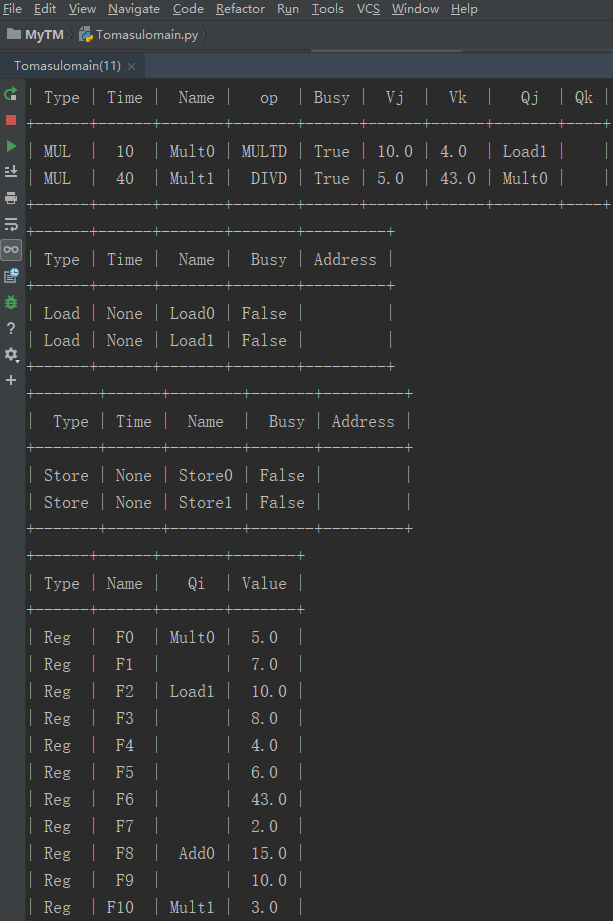
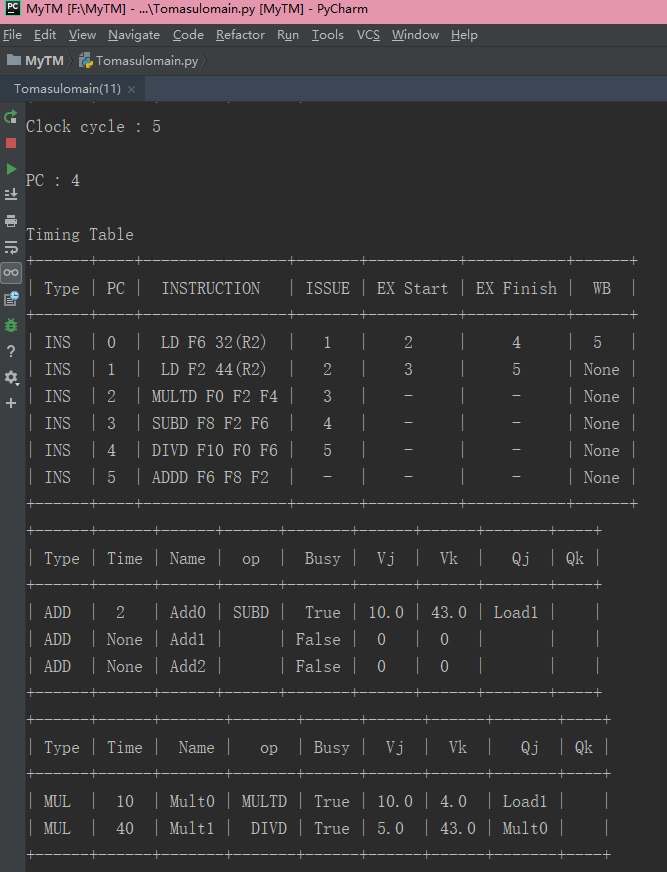


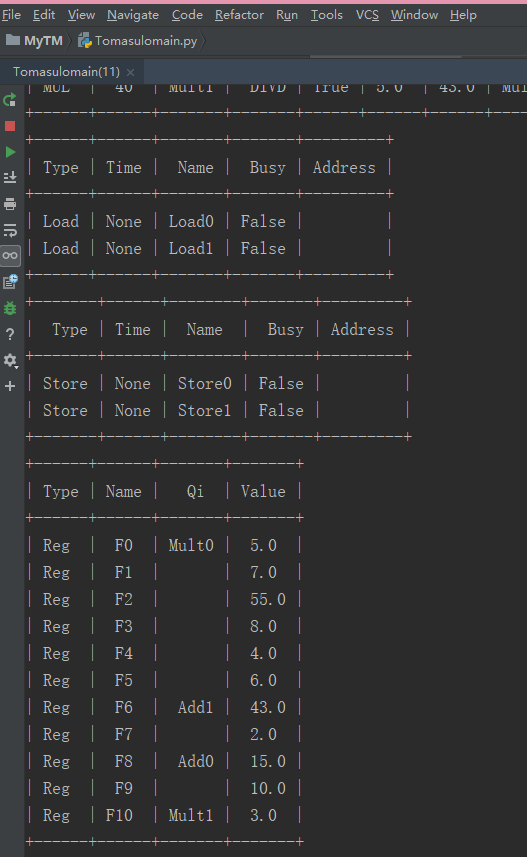
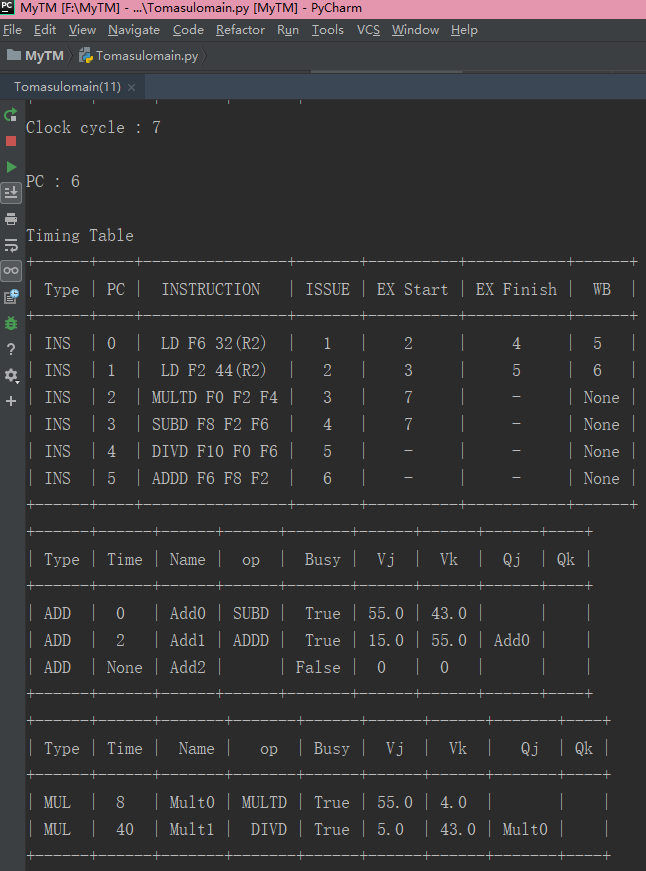
According to the initial Info, the final result should be 58 cycles total, and the final Register value should be [220.0|7.0|55.0|8.0|4.0|6.0|67.0|2.0|12.0|10.0|5.1]. And some key points’ Screenshots as follows:



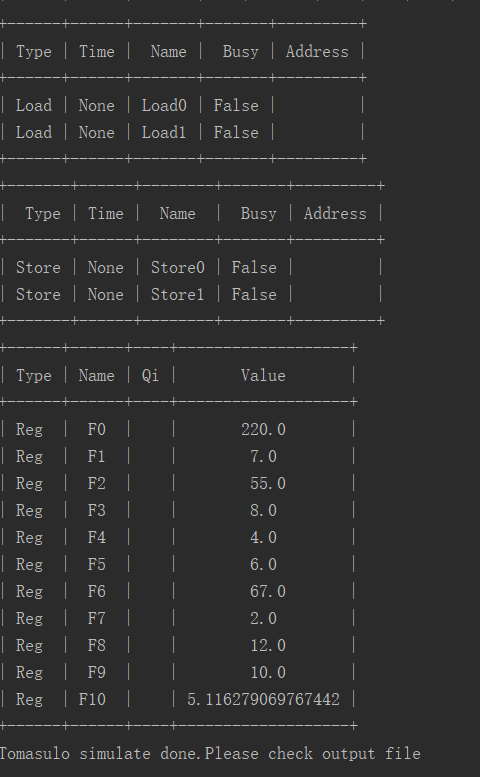
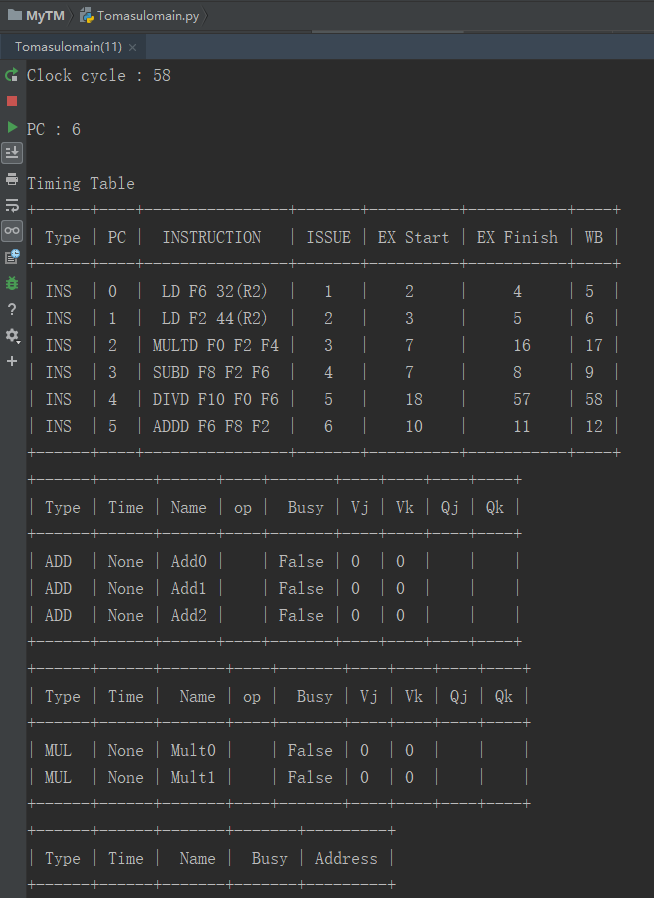
Cycle 1

 Cycle 4

 Cycle 5



Cycle 7

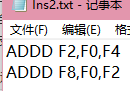


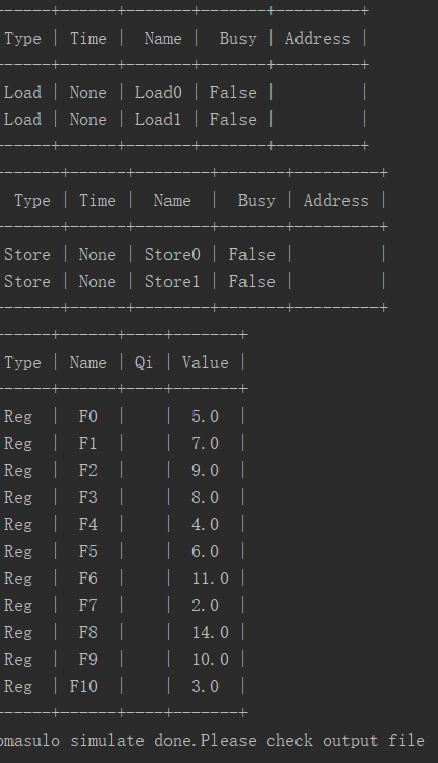
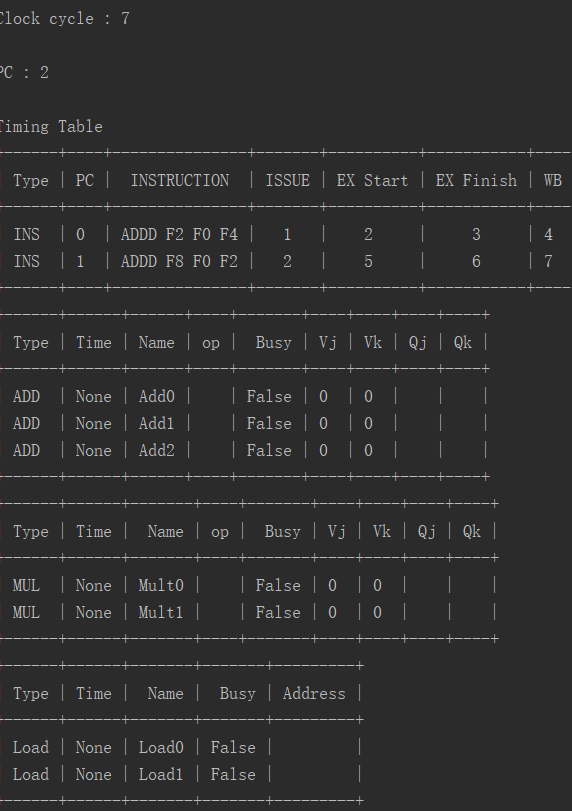
Cycle 58

From the screenshots of cycles 1, 4, 5, and 7, it is obvious that the system satisfies the one cycle delay of the write-back operation and the delay when both operands are available. According to the screenshot of the 58th cycle, it can be seen that the final result and the life cycle of each instruction are in line with theoretical predictions.

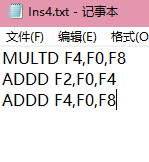
Then, I used some short small examples to verify whether three data hazard can be solved (examples also from the lecture slides),Due to they are short so the report focuses on the final result, and detail can be found in output txt files.

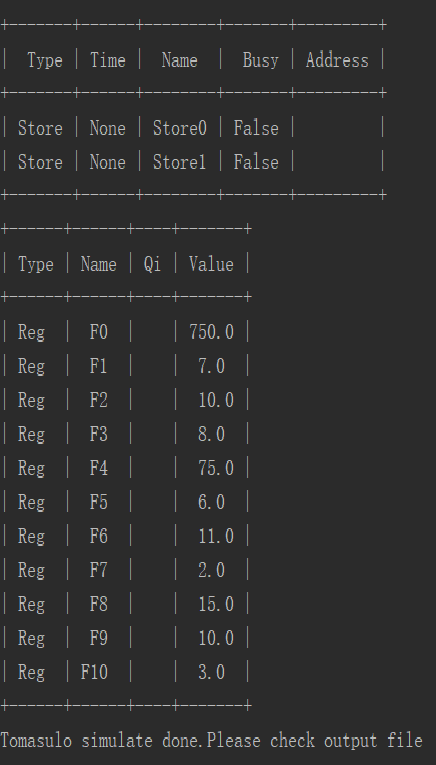
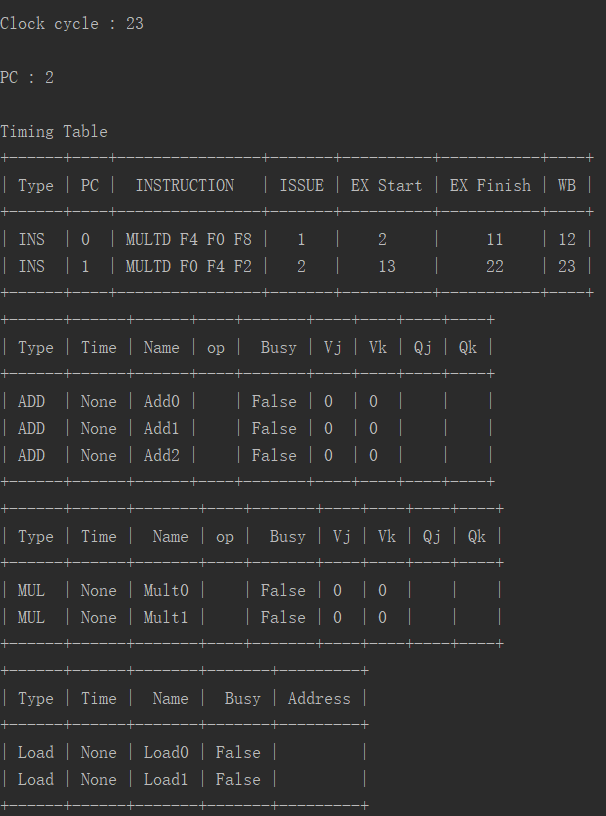
RAW：



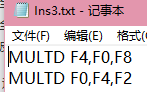


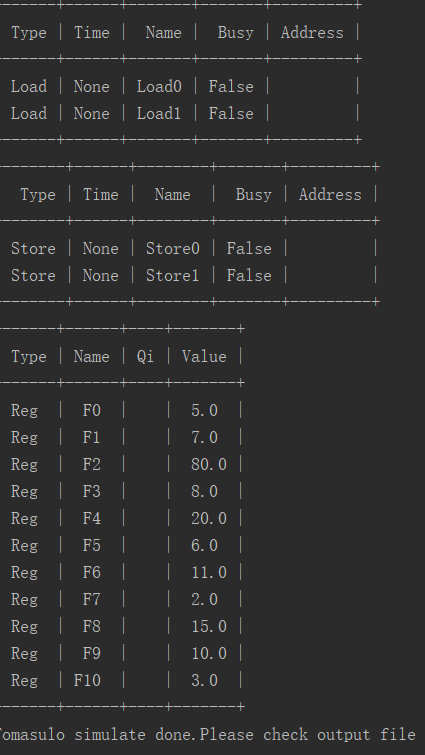
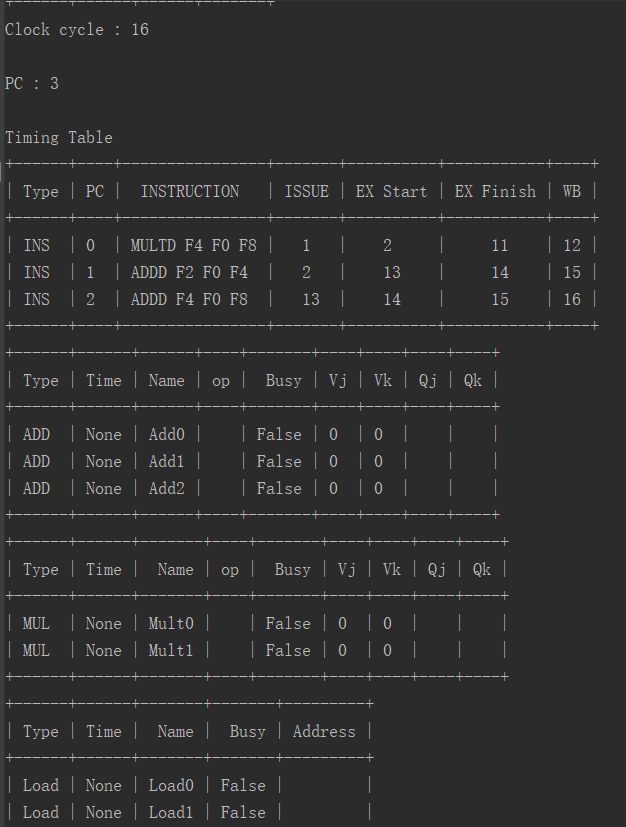
WAR：





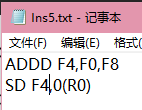
WAW：

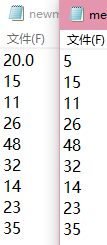




From the above results, it can be seen that the module function of the algorithm runs successfully, and could overcome three hazards.

The last point is about the SD command test:





Obviously, the Mem of R0 was successfully stored the new calculated value of F4——20.Thus the Store function is successfully realized.

# 4.Appendix(Source Code)

Because the code are too long to put in the appendix, I Separate the source codes used in the experiment in different word documents. There are 5 such documents ：**instruction\_queue.py、reservation\_station.py、LoadandStore.py、register.py、Tomasulomain.py.**