Cache Simulator

**PART A) Reflection**

Group Members: Sean Atangan

Activity Log:

|  |  |
| --- | --- |
| April 28th | Fix Project 2, Clean code, Clean output, Develop Plan for Cache Simulator |
| April 29th | Write program intro, obtain user input, create bit configuration variables to match needed cache info (offset bits, setID bits, block#, tag, block age) |
| May 2nd | Build function cacheSend() to be called from LW and SW sub-functions, passing variables to be used for cache. Calls a cacheConfiguration function depending on user input. |
| May 4th | Added needed functions for test programs: Ori, Addu  Debugged the following functions: Slt, Beq, |
| May 5th | Built cacheConfig1 and cacheConfig2 functions, debugged cacheConfig2 |
| May 6th | Built cacheConfig3 and cacheconfig4 functions, Inserted print statements for user specified Detailed display view |
| May 7th | Debugging cacheConfig3, cacheConfig4, LRU functionality |
| May 8th | Write report |

**What are some features that were useful?**

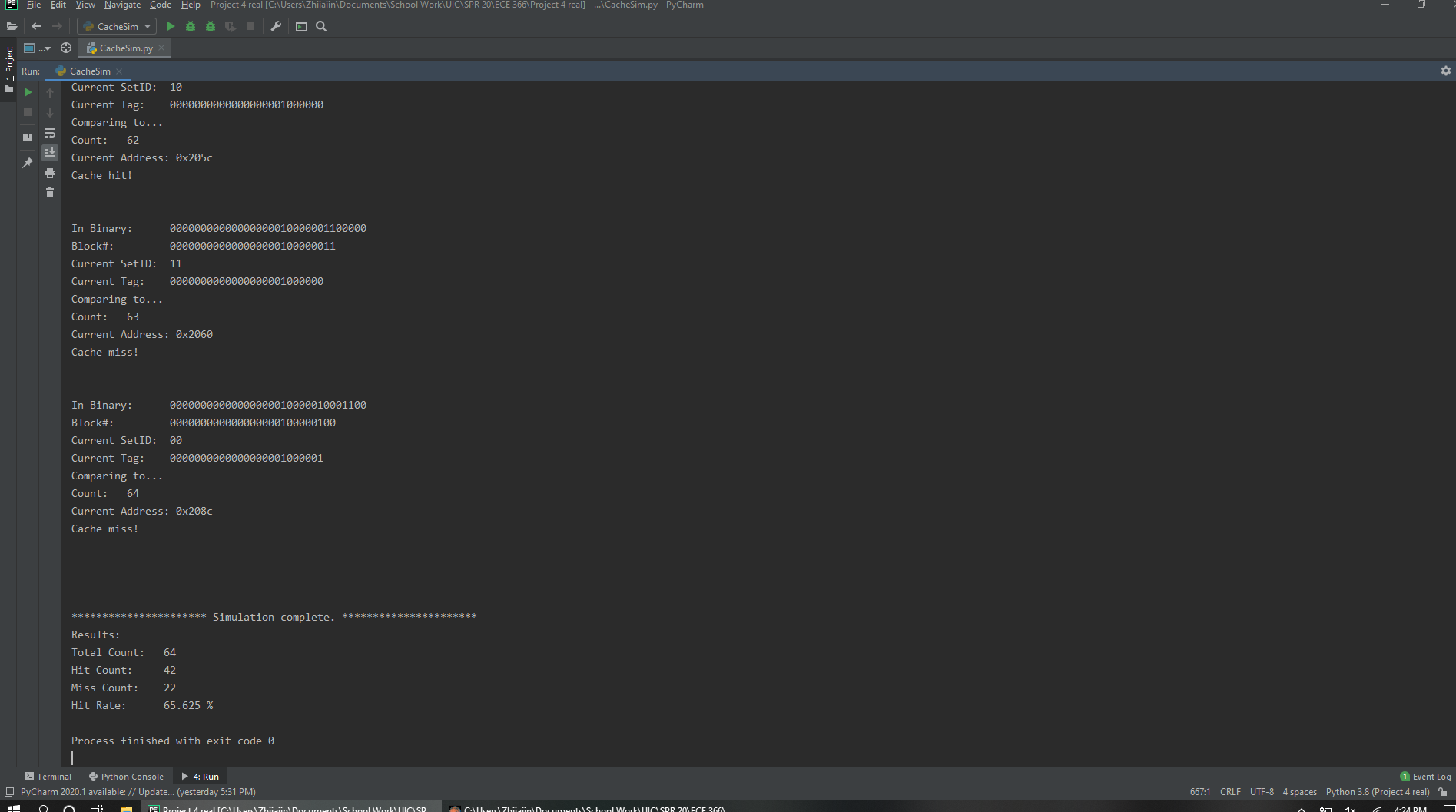
I found that the python debugger pdb.set\_trace() was extremely useful for finding where my code was breaking or when cache information did not match the data cache simulator in MARS. I also really like how python utilizes indentations to determine scopes, which makes coding easier and cleaner that writing in C or C++.

**What would I advise other students about the projects in this course?**

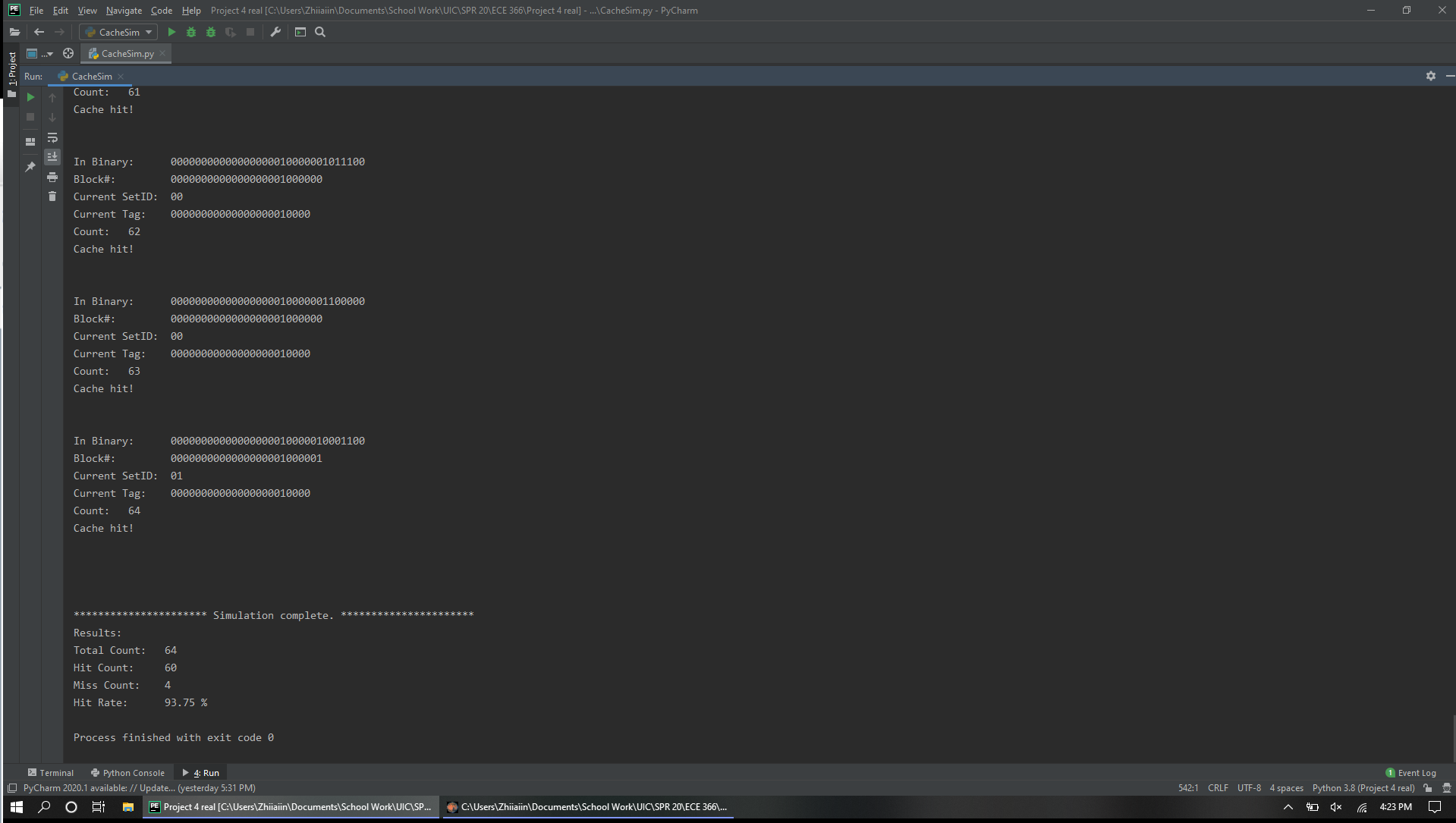
I would recommend students to dig really deeply into the course material especially on days of which the content is relevant to the project. This may be already obvious to some, but really understanding how the machine you are building works, down to the smallest details, will help with the efficiency of your code as well as allow for other on the spot learning that may arise from coding in a new programming language. Otherwise, these projects are as fun as they are challenging, and they do a great service of teaching one how to think like an engineer. They also stress how crucial it is in the real world to be able to work in a team, and can uncover how difficult it can be to work on a real life problem all on your own.

**PART B) Showcase**

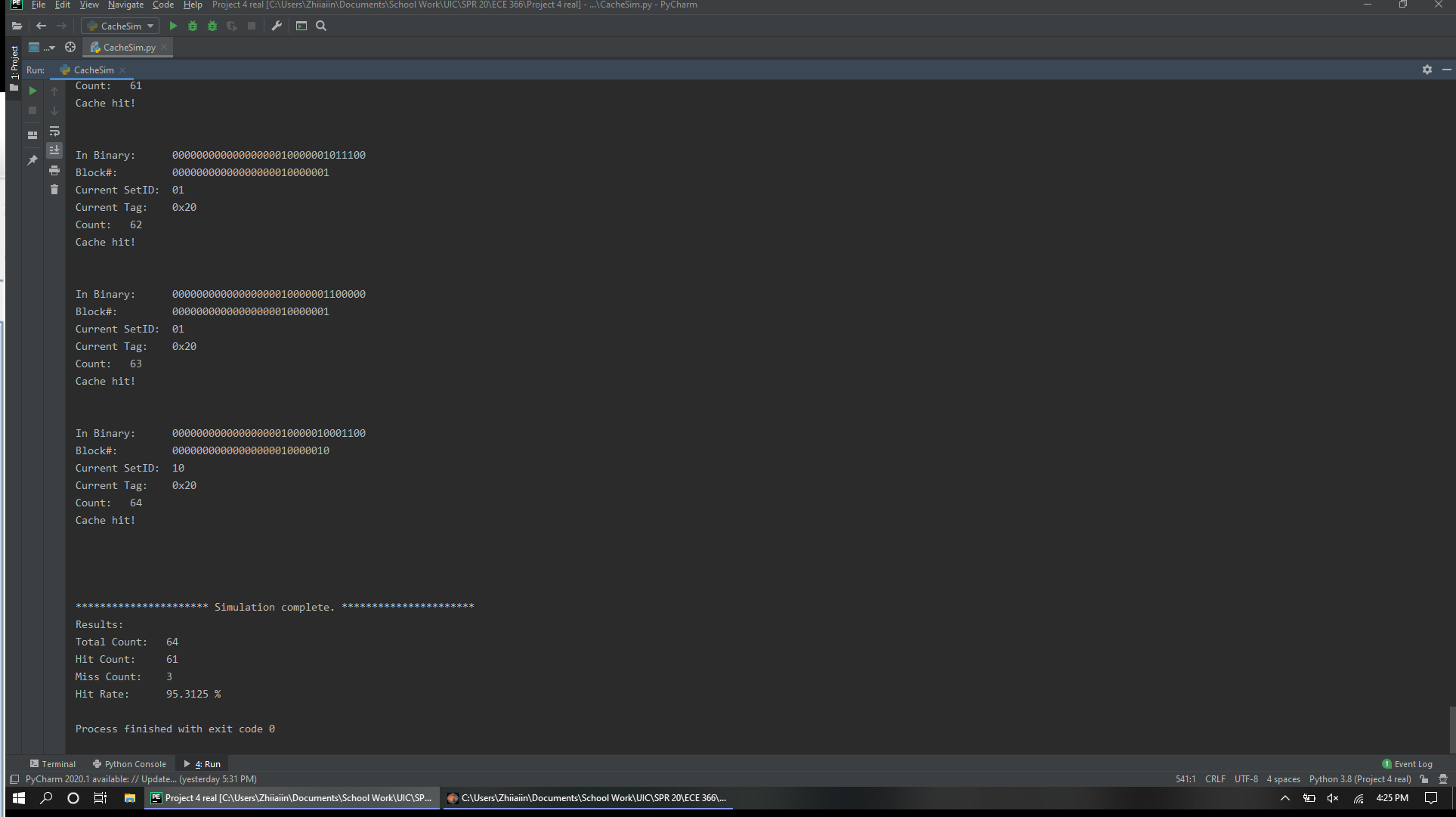
Config 1, detailed mode



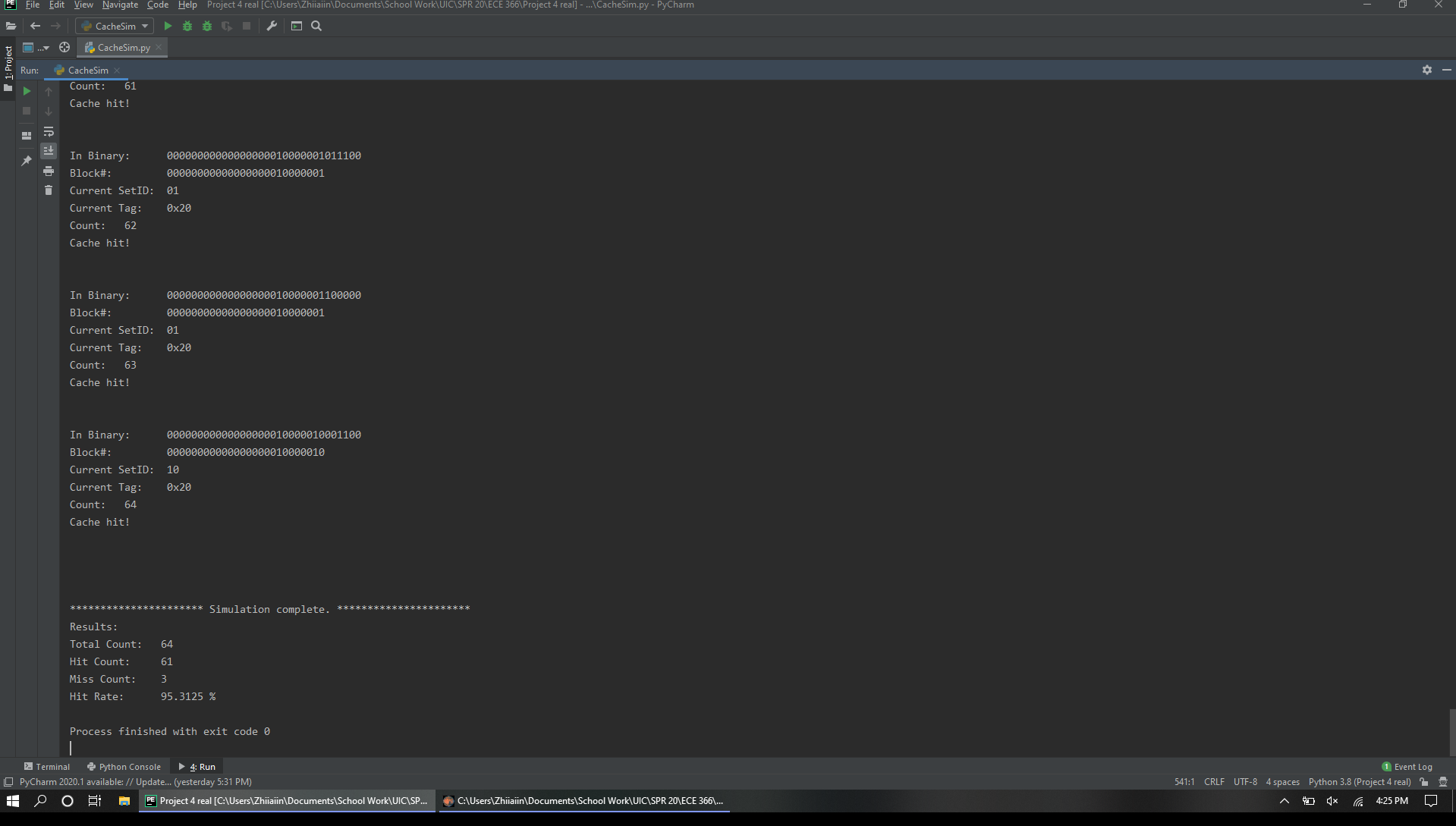
Config 2, detailed mode



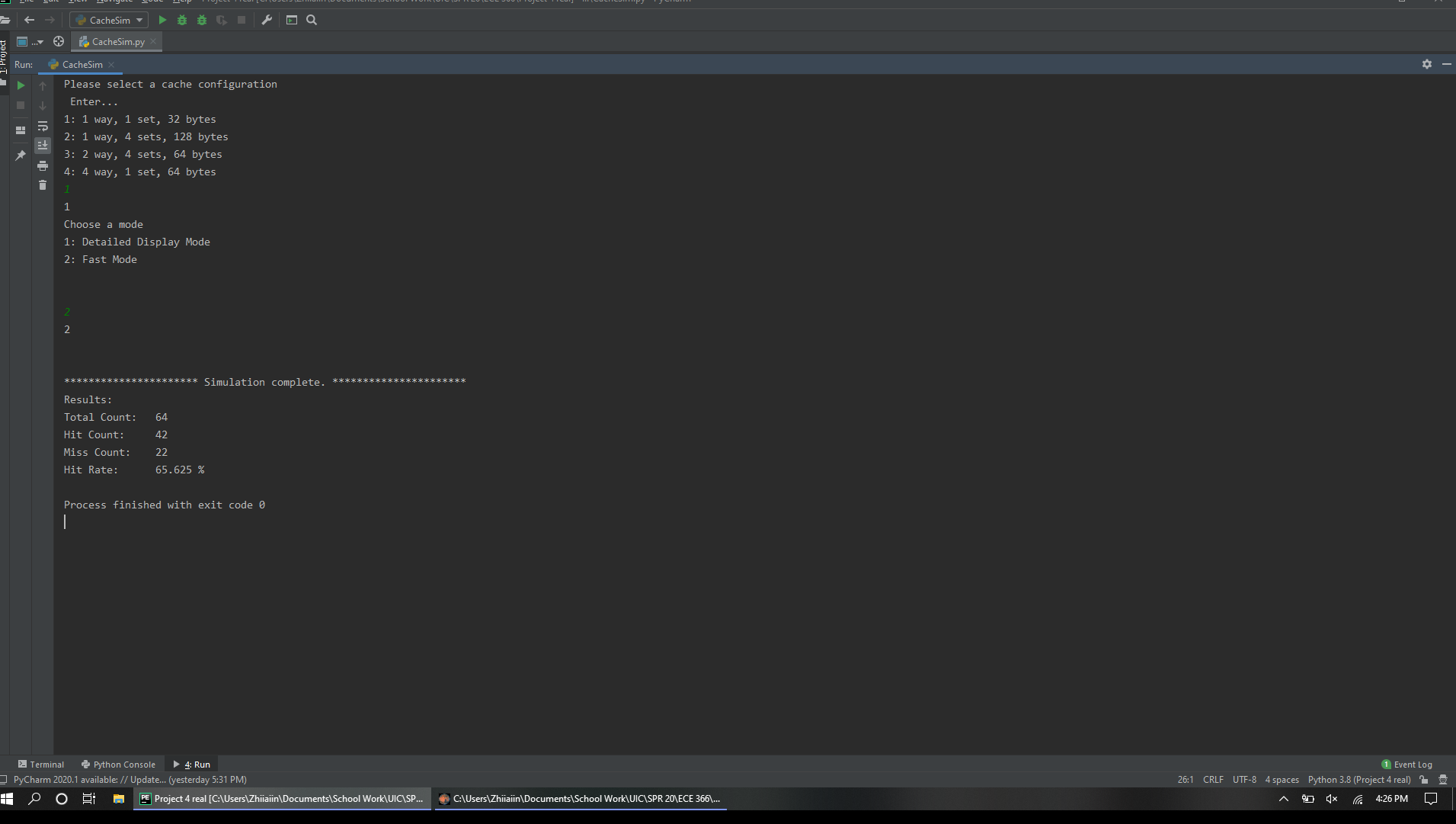
Config 3, detailed mode



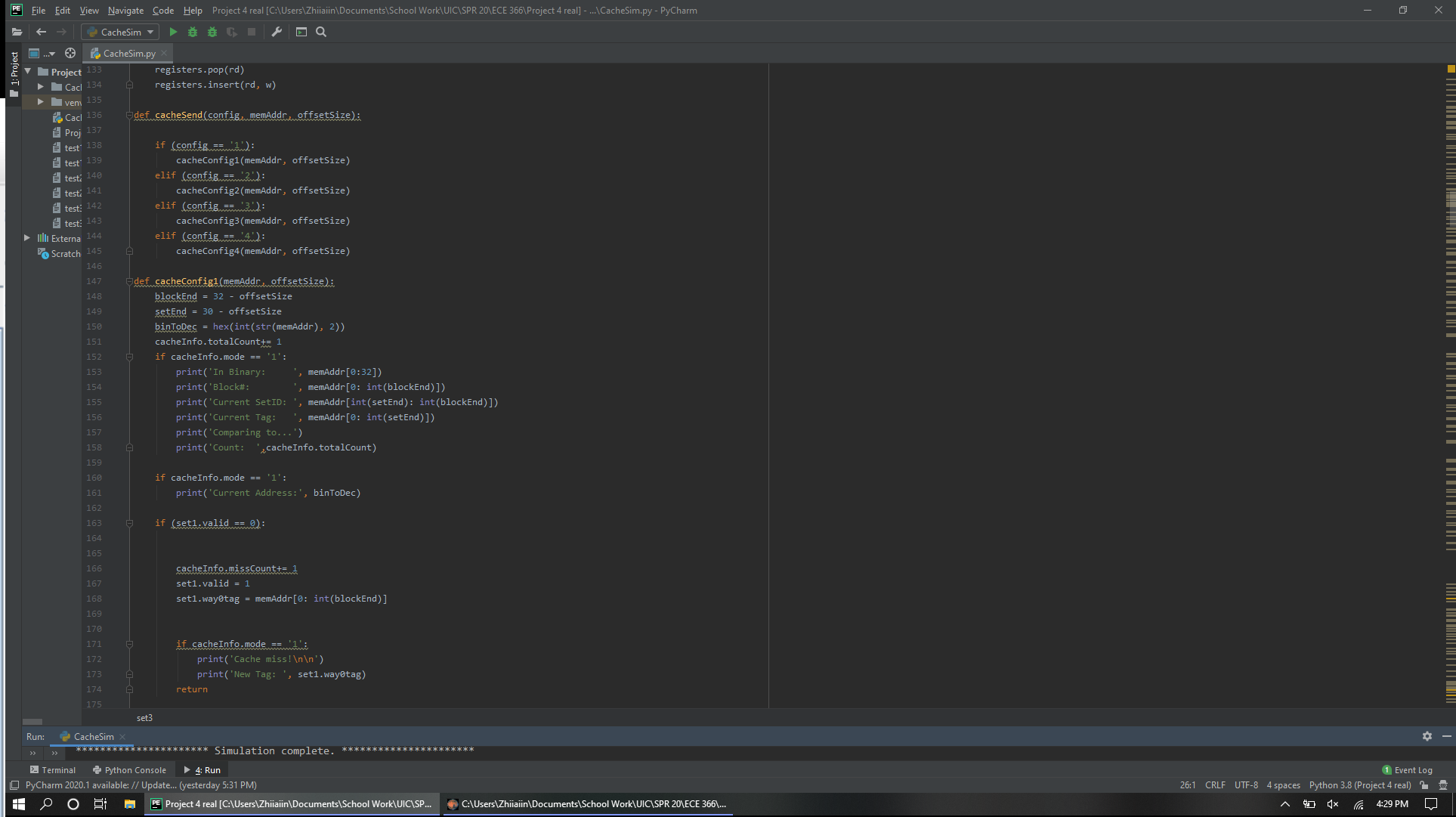
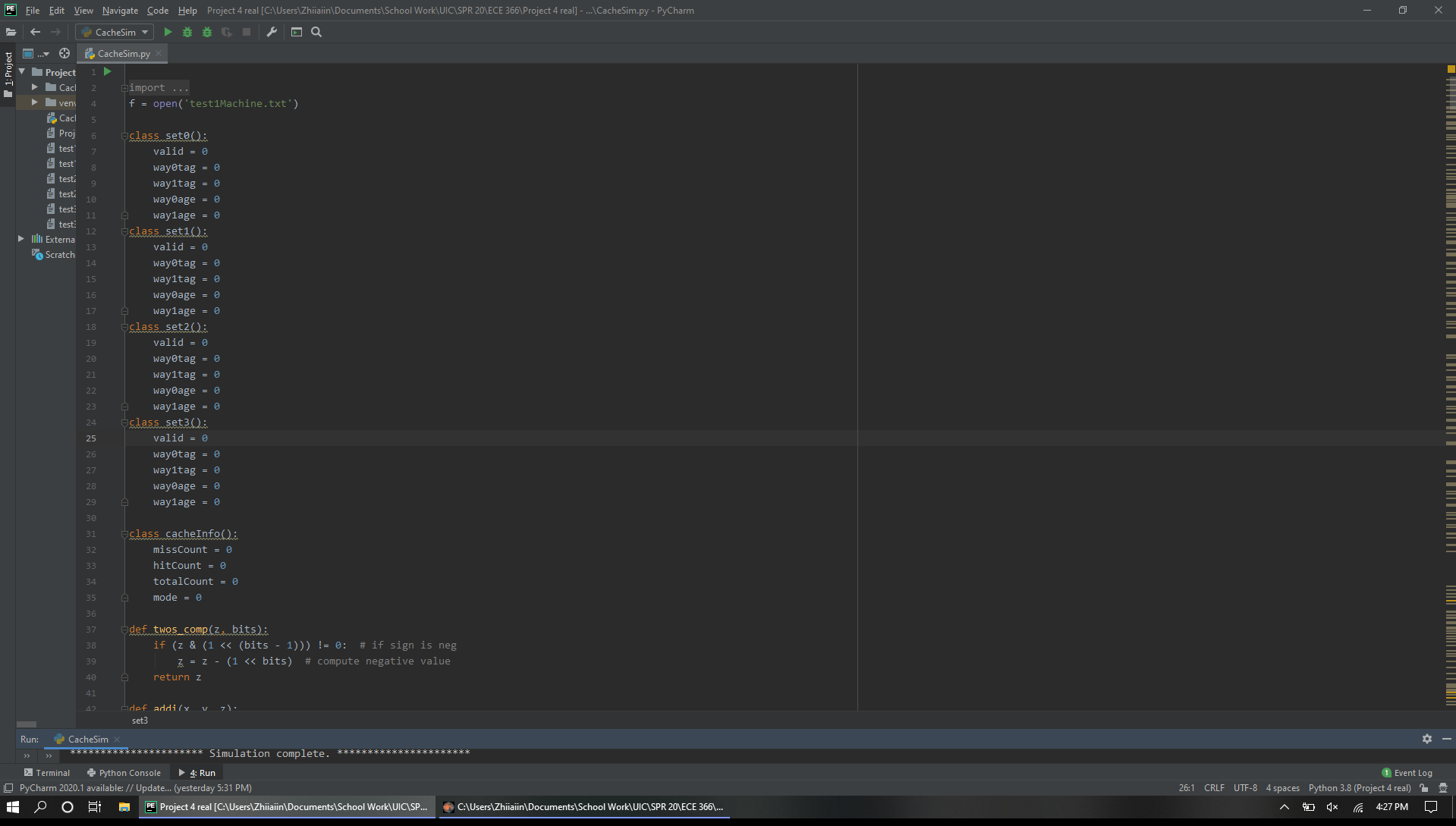
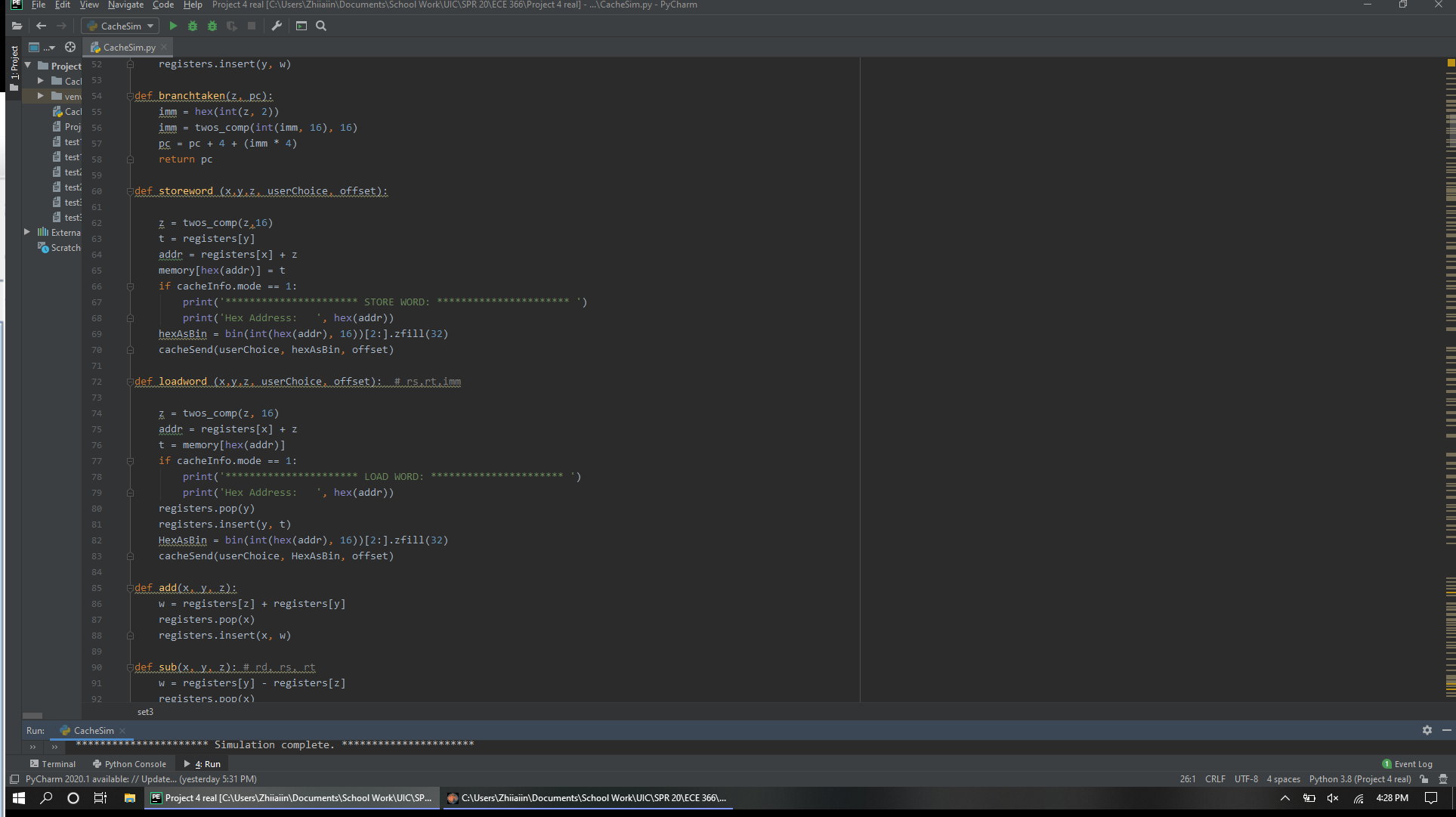
Config 4, detailed mode



FastMode (Similar for all)



Other Code Snippets



**Overall Conclusions**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test 1** | **Test 2** | **Test 3** |
| **Config 1 Hit Rate** | **65.63%** | **72.88%** | **80.95%** |
| **Config 2 Hit Rate** | **93.75%** | **93.22%** | **93.65%** |
| **Config 3 Hit Rate** | **95.31%** | **94.91%** | **95.24%** |
| **Config 4 Hit Rate** | **95.31%** | **94.91%** | **95.24%** |
|  |  |  |  |

I found that either Cache Configuration 3 of 4 yields the best performance. This data implies that more parameters are important for maximizing performance, and that different parameters for cache need to be customized depending on what hardware will use it. The main factors seem to be how much of the block number bits do you want to conserve verses how efficient do you want memory to be accessed. These are things to consider when working on CPUs and determining which configurations will optimize performance, at the same time ultimately save the company the most money during production.