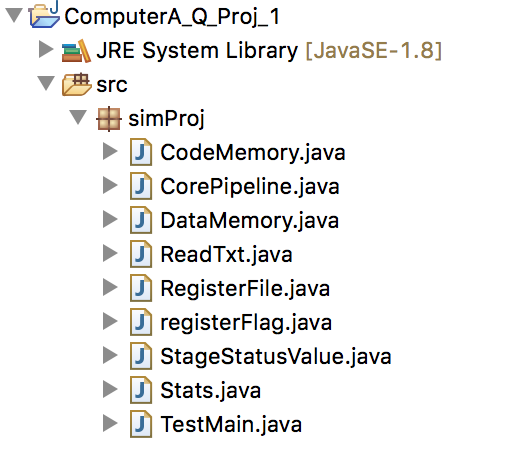
**Design Documentation**

**Computer architecture and organization (CS 520)**

**Summary:**

I use java to finish project 1. Main reason for selecting java is that I want to improve my java skills. Actually, I believe C language is the best choice to realize this project. Because C is a procedure-oriented language, it is good at realize some design relate to flow control. Java is object-oriented language. Use java to realize pipeline seems a little weird. So, in my design, I build a lot of functional classes, and try my best to reduce quantity of instances. I believe if there were not so many real objects, no need to create a lot of instances, only some methods were used from them.

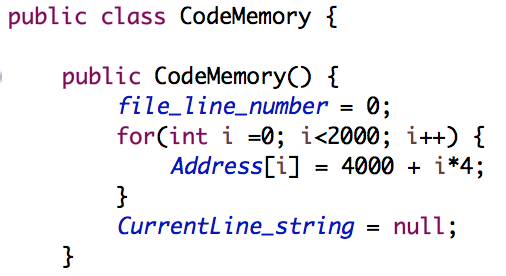
The structure in my design:



**CodeMemory:**

Set up a integer array Address[2000] to simulate code address in computer.

Address example: 4000, 4004, 4008, 4012….



**CorePipeline:**

Each stage’s functions are built in CorePipeline.

main() is in TestMain.java.

The flow of pipeline is run in main(), for example, passing instruction’s status from stage to stage.

In each stage, the specific function in CorePipeline will be called:

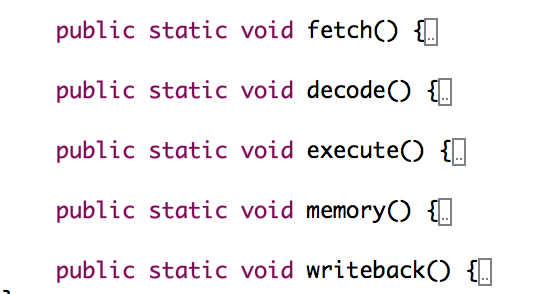
fetch(): read the instruction from input.txt.

decode(): decode the instruction and fetch relate data. (dependence is set in main().)

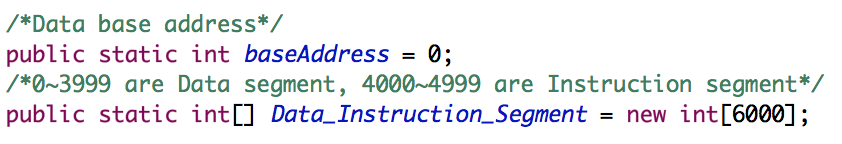
execute(): calculating work and.

memory(): load and store work from memory.

writeback(): write register and set PSW.



**DataMemory:**



**ReadTxt:**

It is a functional class, read the information from input.txt and translate them in instructions, saving in a array.

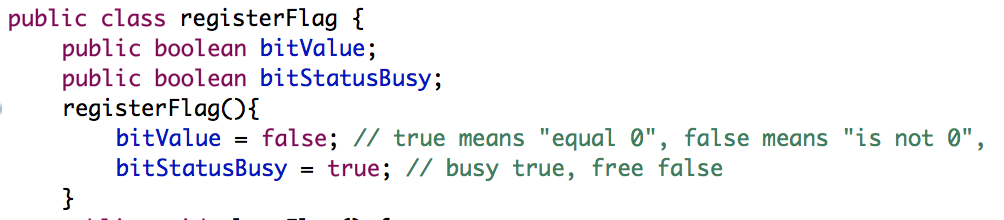
**RegisterFile:**

Set an integer array for registers.



**registerFlag:**

Set register flag class.

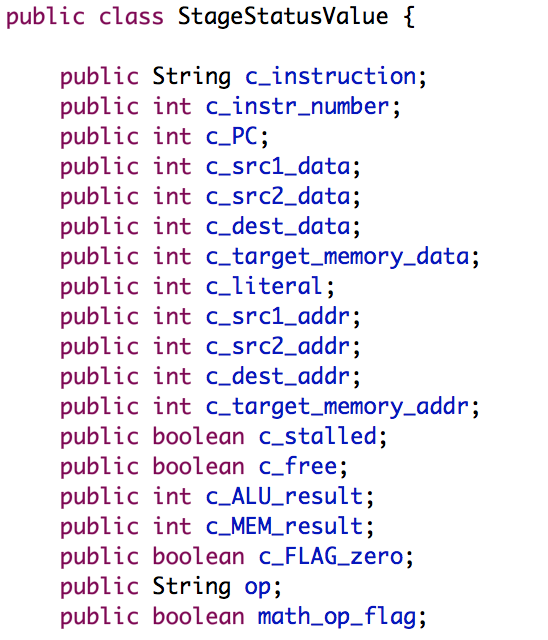


**StageStatusValue:**

This class is for keep status information in each stage.

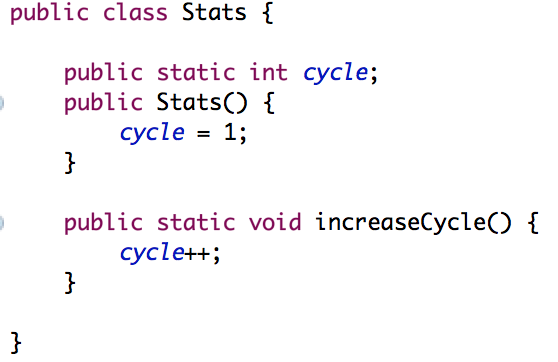
In main(), these status information is passed from stage to stage.

This class is reset status methods.



**Stats:**

Cycle class.



**TestMain:**

This is the main class which contains pipeline flow control (interlock dependency, stage management, timing control about passing status from stage to stage…), Simulator Commands control and display methods.

**Addition information from requirement document:**

1. **register file**

RegisterFile & registerFlag.

1. **Instructions**

Instructions saving in ReadTxt.



1. **code and data memory**

CodeMemory & DataMemory

1. **core pipeline**

CorePipeline

1. **flow/algorithm in each stage**
2. In Fetch stage: Each cycle fetching a new instruction if no block from DRF. Start setting instruction status from this stage, such instruction and instruction address and instruction number.

Fetch method is called from CorePipeline class.

1. In DRF stage, if no flow dependency, source data can be retrieval. In order get the data from register, instruction status should be pass to DRF method in CorePipeline class. Then DRF stage in main() can get the update data from relate register.
2. EX stage contains EX\_Integer and EX\_MUL/MUL2 stage. Instruction status should be pass from main() to EX method in CorePipeline class. After calculating, main() can get the update status about the instruction. PSW data should be add in status.
3. In MEM stage, instruction status should be pass from main() to MEM method in CorePipeline class. After operation, main() can get the update status about the instruction.

Instruction status can get some new data such as data from memory.

1. In WB stage, instruction status should be pass from main() to WB method in CorePipeline class. After operation, relate register can be updated, including PSW.
2. Code structure in main() is important. Program should run as the flow: WB 🡪 MEM 🡪 EX 🡪 DRF🡪 FETCH.
3. **interlocking logic**

Only flow dependency need to be considered. Some key points as below:

1. When instruction enter DRF, compare the DRF src1/src2 to EX dest and MEM dest. If same, produce block.
2. When instructions come in EX MUL2 and EX Integer stage simultaneously, instruction in EX MUL2 has the higher priority entering MEM.
3. PSW contains two parts: value and status. Each arithmetic instruction in my design has ability to set PSW status busy. If arithmetic instruction wants to set PSW free when it in WB stage, something should be check: Are there any arithmetic instructions in DRF & EX & MEM stages? If yes, the instruction in WB stage cannot set the PSW free. Vice versa.