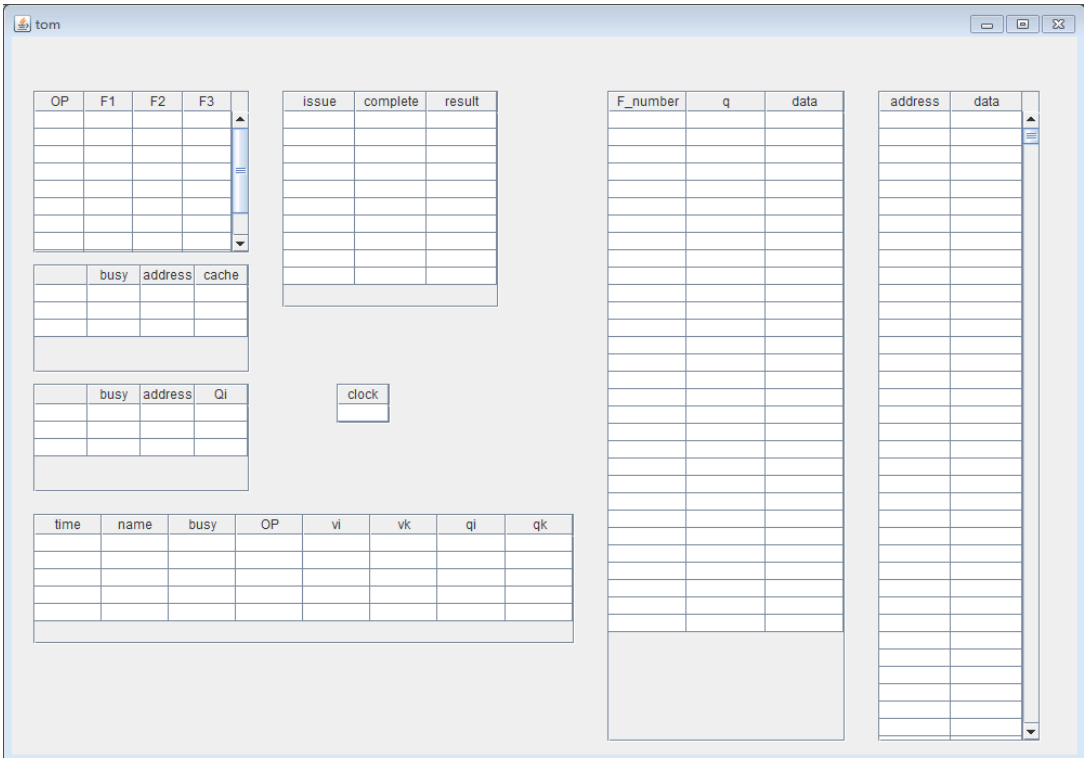
【实验说明】

本实验的实现方式是用的 java 语言实现的。其中 data.java,datainit.java 分别用于存储数据和数据的初始化。tomasulo.java 实现了 tomasulo 算法（单步），tommain.java 实现了整体的调度并且生成界面。

实验结果界面如下：

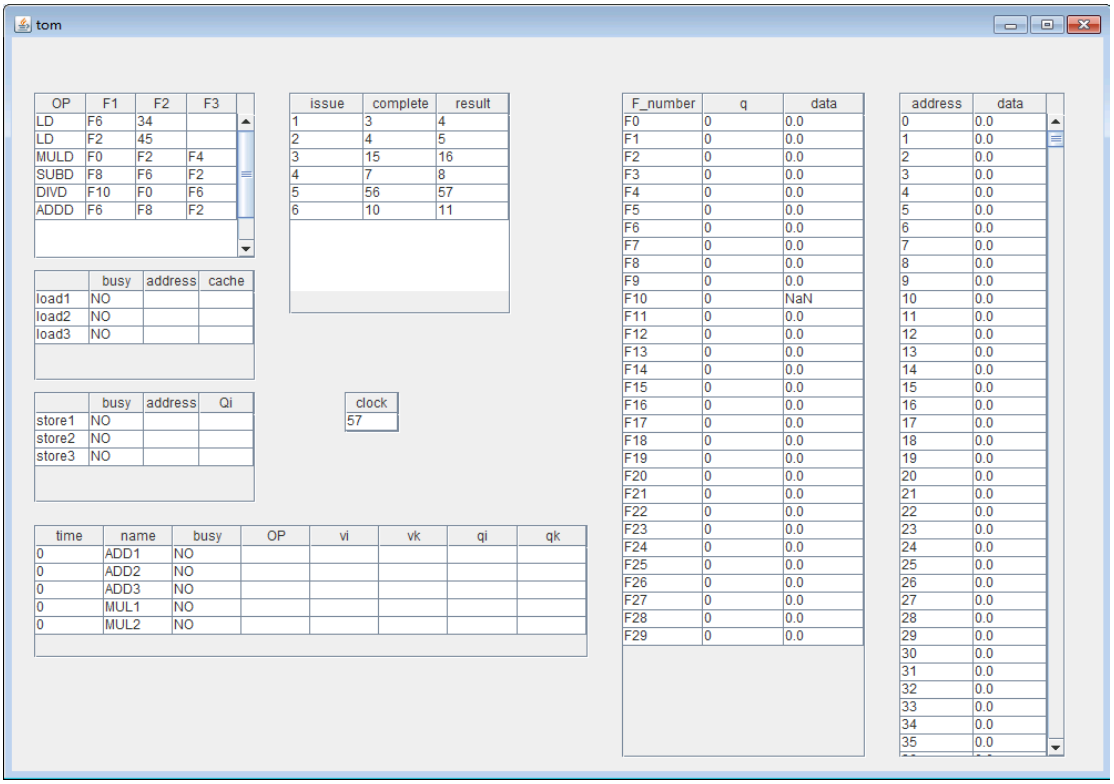


最左侧从上到下分别为：指令，load，store，reservation station。

中间上方为：指令执行时空图，其中当指令的执行阶段完成时 complete 才会出现数据。

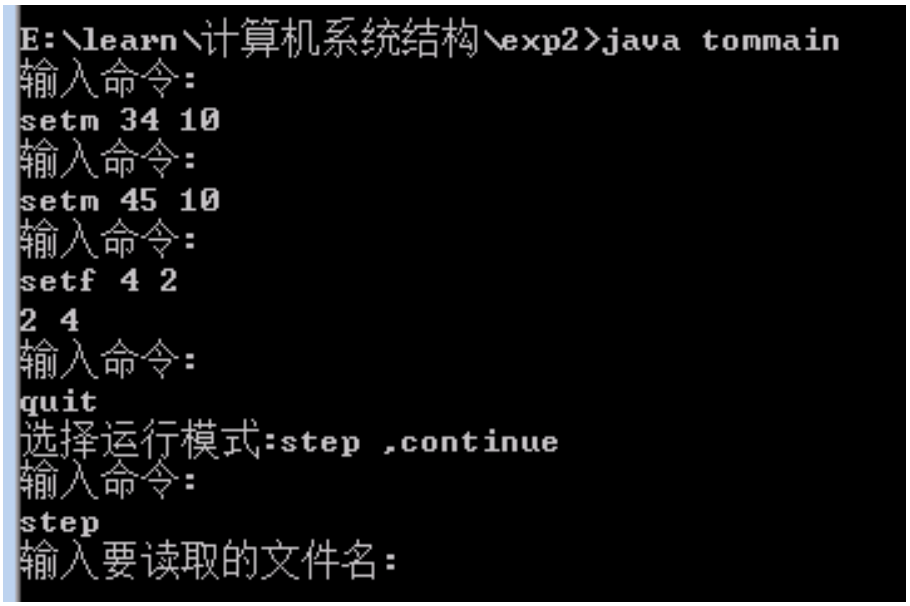
右侧两个分别为：寄存器和内存。其中寄存器的 q 代表在执行时，此寄存器是否被占用。

用 order.txt 文件中的指令测试结果如下：



在运行程序时，建议用 cmd 进行运行，在 cmd 中可以预先设置寄存器和内存的值：

在 cmd 下执行 javac tommain.java 和 java tommain 得到如下：



其中输入命令是指，setm 和 setf，分别为设置内存中的值和寄存器中的值

setm a b,为将 b 的值写入地址为 a 的内存中。

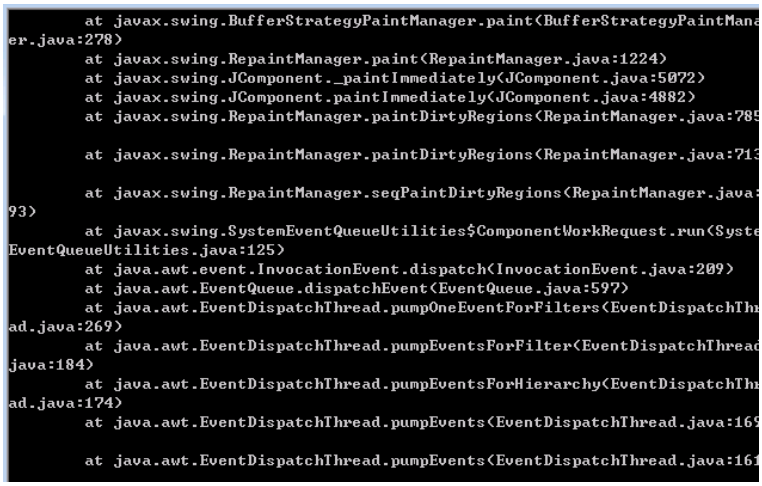
setf a b 为将第 a 个寄存器的赋值为 b。

输入 quit 后结束赋值。

step 和 continue 分别为单步和连续的执行方式。

再输入指令所在的文件的名字即可。

注：在执行过程中可能会出现如下情况


at javax.swing.BufferStrategyPaintManager.paint(BufferStrategyPaintMana
er.java:278)
at javax.swing.RepaintManager.paint(RepaintManager.java:1224)
at javax.swing.JComponent._paintImmediately(JComponent.java:5072)
at javax.swing.JComponent.paintImmediately(JComponent.java:4882)
at javax.swing.RepaintManager.paintDirtyRegions(RepaintManager.java:785
at javax.swing.RepaintManager.paintDirtyRegions(RepaintManager.java:713
at javax.swing.RepaintManager.seqPaintDirtyRegions(RepaintManager.java:
93)
at javax.swing.SystemEventQueueUtilities$ComponentWorkRequest.run(Syste
EventQueueUtilities.java:125)
at java.awt.event.InvocationEvent.dispatch(InvocationEvent.java:209)
at java.awt.EventQueue.dispatchEvent(EventQueue.java:597)
at java.awt.EventDispatchThread.pumpOneEventForFilters(EventDispatchTh
ad.java:269)
at java.awt.EventDispatchThread.pumpEventsForFilter(EventDispatchThread
java:184)
at java.awt.EventDispatchThread.pumpEventsForHierarchy(EventDispatchTh
ad.java:174)
at java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:169
at java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:161

这个对程序没有影响。