Tomasulo 模拟器实现

王拓为 2018011917

一、运行方式

模拟器使用 python3 完成。

首先执行如下命令配置环境：

> sudo pip3 install "web.py==0.62"

然后执行如下命令运行模拟器：

> python3 main.py [port]

其中 `port` 可缺省，默认为 8080 端口；

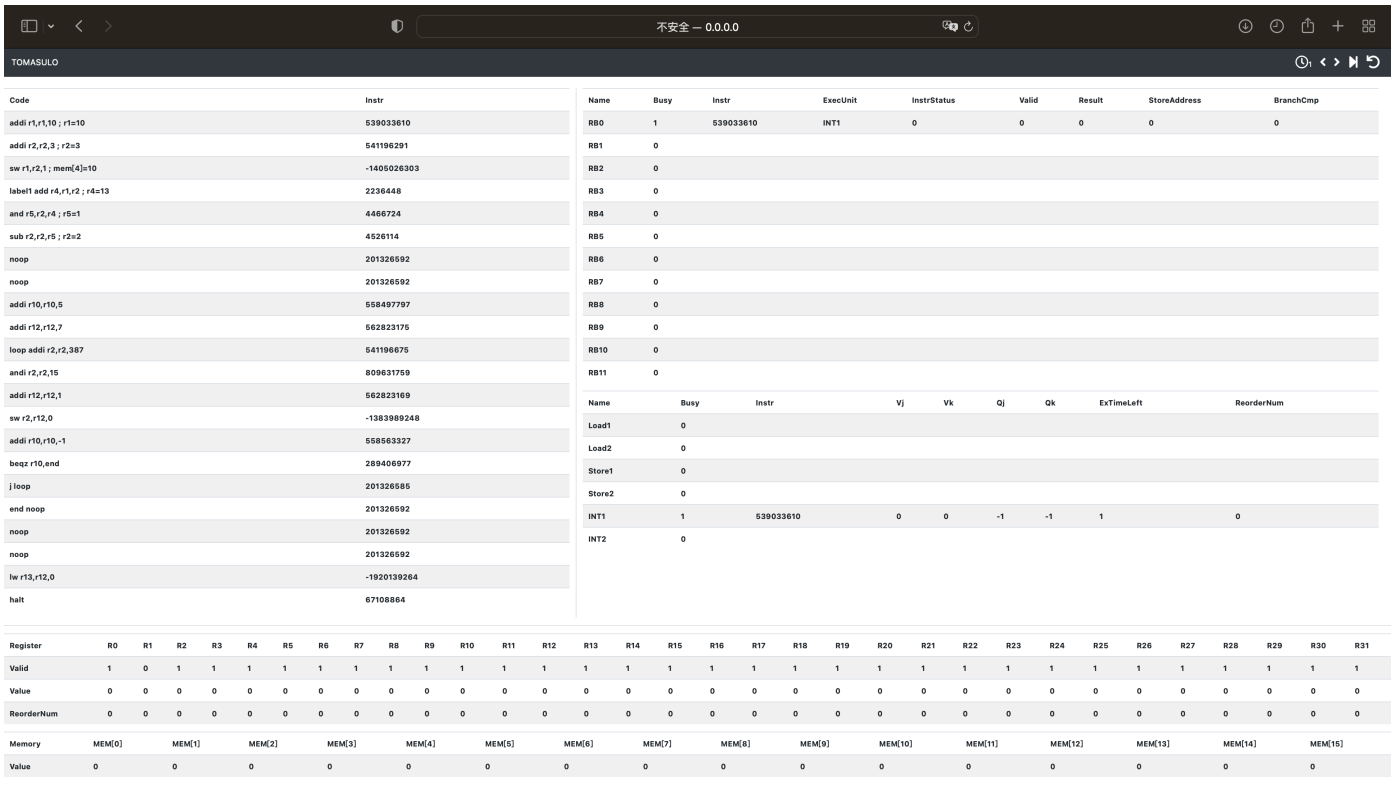
最后在浏览器中打开 `https://localhost:[port]` 即可。

二、前端效果

前端使用 Bootstrap4 实现。

页面显示了指令、ROB、保留站、寄存器和内存的完整信息。

支持单步执行，单步回退，运行至程序结束和重置功能，效果图如下：



三、后端设计

后端分为汇编器、模拟器和服务器 3 部分。

汇编器部分负责将程序由汇编代码转为机器代码，位于 `assembler.c` 中。

由于使用了 C 代码进行编写，文件的读写和字符串的处理相对繁琐，但实现思路较为简单，即根据指令格式，将读入的字符串翻译为对应的机器码。其中有两点需要注意：

一是汇编代码中标签的处理，我的做法是通过两次遍历，在第一次遍历中记录标签及其对应地址信息，供第二次遍历真正进行代码转换时使用；

二是由于模拟器框架中使用4字节整数表示指令，因此需要将01机器码转化为对应4字节整数。

模拟器部分负责根据 Tomasulo 算法模拟机器代码的执行，位于 `tomasulo.c` 中。

具体地，模拟器的每个时钟周期需要：

1. 首先，确定是否需要清空流水线或提交位于 ROB 的队首的指令。对于条件跳转指令默认跳转不成功，如果预测错误，则清空流水线，并设置新的 PC 值为跳转地址。如果不需要清空，且队首指令能够提交，则更新状态，修改相应寄存器和内存。完成清空或提交操作后，释放保留站并更新队列的首指针。
2. 其次，对所有已发射的指令进行处理。对于发射状态指令，检查两个操作数是否都已经准备好，如果是，将指令状态修改为执行状态；对于执行状态指令，将剩余执行时间递减，当执行完成时，将指令状态修改为写结果状态；对于写结果状态，将结果复制到正在等待该结果的其他保留站中去，同时将结果保存在 ROB 中的临时存储区中，释放指令占用的保留站, 将指令状态修改为提价状态。
3. 最后，检查是否能够发射一条新的指令。首先检查 ROB 中是否有空闲的空间，如果有，检查所需运算单元是否有空闲的保留站，如果依然有，则发射指令。在发射指令时填写对应保留站和 ROB 的内容，如果指令在提交时会修改寄存器的值，还要对相应寄存器进行预约。

需要说明的是 Qj 和 Qk 中存储的是对应 ROB 编号，且值为 -1 时表示对应的 V 有效（避免与 0 号 ROB 冲突）。

服务器部分负责将模拟器的输出文件转化为前端输入，位于 `main.py`。

为了便于转化，模拟器的输出格式与服务器进行了约定；服务器与前端使用 json 文件交互。