设计实现 Tomasulo 算法模拟器实验报告

计 42 班 杨天龙 2014011310
计 42 班 杨松涛 2014011316
计 45 班 徐子南 2014011405

算法介绍

本次作业，我们实现了课本上的 tomasulo 算法，以下对该算法进行简要描述。

tomasulo 算法的核心是采用某种方法对寄存器进行换名，从而消除一些指令运行时的冲突。

一条指令执行前，他将会对自己将要写入的寄存器进行“预定”，声明运算结束之后自己将对这一寄存器进行修改。而只有当某一指令的所有需要的寄存器都在该指令进行“预定”前尚未有人预定，这一指令才得以开始。

而一旦某一在保留站（已流出但尚未运行的指令的储存处）中的指令所有被预定的寄存器已经就绪，那么他便随时可以开始运行。

一条指令结束后，他将会通过 CDB（公共数据总线）把结果发送

到所有需要自己之前“预定”的数据的地方，包括某一保留站中的指令和某个寄存器。但是根据助教的说法，CDB 可以视为无限带宽，因此也实现了这一版本。

以下对我们实现的一些细节进行说明：

由于作业要求的乘除法器的说明有不清楚之处，我们在进行了讨论和对助教的询问之后，决定采用如下实现方法：浮点乘除法器作为一个整体包装，不做流水处理，乘法需要 10 个周期进行运算，而除法需要 40 个周期。而浮点加法器依然是一个 2 周期的流水线结构。

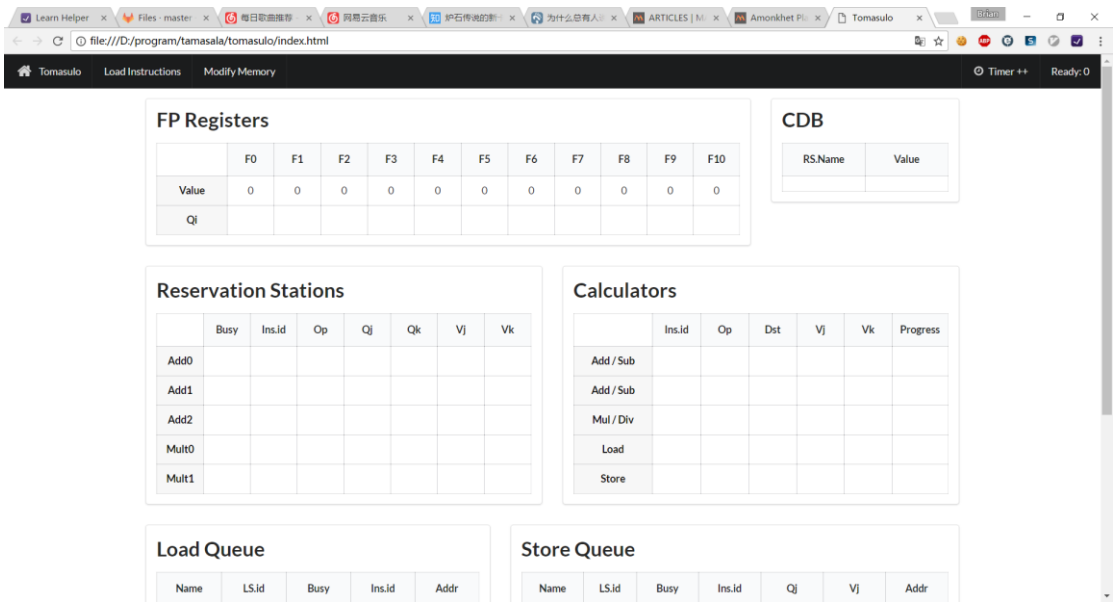
我们讨论认为，CDB 应该是一条串行总线，一次只能发送一条数据，我们在 CDB 中加入了仲裁机构，使得其会按照乘除->读写->加减的顺序进行发送。在有数据等待时，我们将会阻塞整个流水。

==========

==========

操作手册

以下配合截图进行说明。



左上角：

Load Instructions 按钮：点击后弹出对话框，在此可以输入由换行符分隔的多条指令进行装载。

Modify Memory 按钮：点击后弹出对话框，在此可以根据地址装载内存。

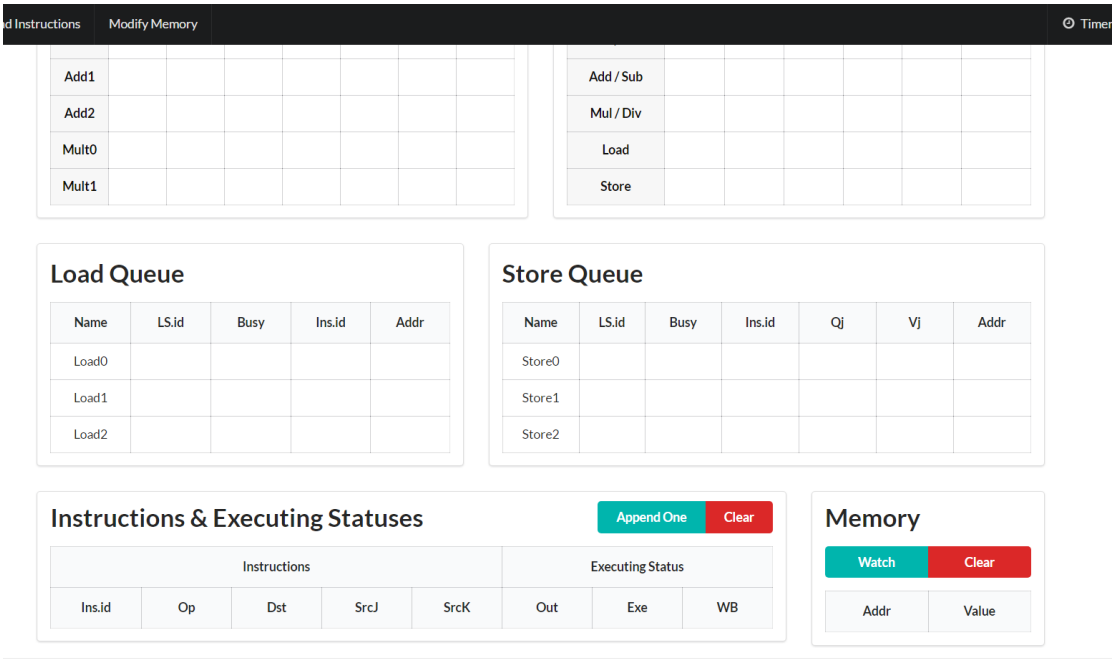
右上角：

Timer 相关（重要）

鼠标移动到该按钮时，将会弹出一个列表，点击列表中的按钮，将改变 Timer 按钮的功能，默认为 Timer++。

点击 Timer 按钮，将运行该按钮的当前功能。

Ready：0 ：冒号前为当前状态，分为‘Ready’（准备就绪）、‘Running’（正在运行）与‘Complete’（运行完成）三种。冒号后为已经运行的周期数。



Instruction 部分的 Append One 按钮：可以在当前列表后加上一条指令。

Instruction 部分的 Clear 按钮：可以清空当前列表。

Memory 部分的 Watch 按钮：可以注册观察的内存地址。

Memory 部分的 Clear 按钮：清空 Watchlist。

==========

界面部件

以下介绍界面中各部件以及显示的各个部分的含义。

1. 指令队列与执行情况

此处存放了所有已经执行、正在执行与尚未执行的指令。

以下介绍各个字段的含义：

Ins.Id 指令的编号，每条指令会按照输入的顺序得到一个唯一的编号。

Op 操作符，可能是‘ADDD’、‘SUBD’、‘MULTD’、‘DIVD’、‘LD’、‘SD’中的一种

Dst 将要写的寄存器，在 store 指令中为地址

SrcJ 将要读的寄存器之一，在 load 指令中为地址

SrcK 将要读的寄存器之二，在 load 指令与 store 指令中没有这一字段

Out 该指令是否已流出

Exe 该指令是否已经进入运算或存取操作

WB 该指令是否已写回（完全执行完成）

例：

Ins.Id, Op, Dst, SrcJ, SrcK, Out, Exe, WB

'1', 'LD', 'F6', '1234', '', True, True, False

'2', 'ADDD', 'F1', 'F2', 'F3', ...

2. FP & Qi 浮点寄存器与 Qi 数组

包含了与寄存器相关的内容。

以下介绍每个字段的含义：

FP.ID 寄存器名称

Value 寄存器中的值

Qi.String 某个保留站/等待队列中的名称，在该项执行完成后，将会接受到该指令的结果作为本寄存器的值

例：

FP.ID, Value, Qi.String

'F0', '1.23', 'Add2'

3. LQ [3] load 指令的等待队列

load 指令的等待队列，长度为 3

以下介绍每个字段的含义：

Ins.Id 指令的 id，与指令队列中的相同字段含义相同

Name 队列中该项的标识符

LDST.Id 用于表示存取操作顺序的 id，每次进行存取操作选择 id 最小的那一项进行操作

Busy 本项目是否正在运行中

Addr 需要获取的数据的地址

例：

Ins.Id, Name, LDST.Id, Busy, Addr

'1', 'Load1', '6', 'yes', '1234'

4. SQ [3] store 指令的等待队列

load 指令的等待队列，长度为 3

以下介绍每个字段的含义：

Ins.Id 指令的 id，与指令队列中的相同字段含义相同

Name 队列中该项的标识符

LDST.Id 用于表示存取操作顺序的 id，每次进行存取操作选择 id 最小的那一项进行操作

Busy 本项目是否正在运行中

Addr 需要获取的数据的地址

Qj 目标寄存器换名后的保留站名，用于接受该保留站发送

的数据

Vj 如果目标寄存器并未被换名，则将其值保存在这里

例：

Ins.Id, Name, LDST.Id, Busy, Addr, Qj, Vj

'2', 'Store3', '1', 'no', '4000', 'Mult1', ''

5. RS 运算保留站

运算操作的保留站，加减法共保有 3 个保留站，乘除法共保有 2 个保留站

以下介绍每个字段的含义：

Ins.Id 指令的 id，与指令队列中的相同字段含义相同

Name 保留站中该项的标识符

Busy 本项目是否正在运行中

Op 运算符

Qj 目标寄存器 1 换名后的保留站名，用于接受该保留站发送的数据

Qk 目标寄存器 2 换名后的保留站名，用于接受该保留站发送的数据

Vj 如果目标寄存器 1 并未被换名，则将其值保存在这里

Vk 如果目标寄存器 2 并未被换名，则将其值保存在这里

例：

Ins.Id, Name, Busy, Op, Qj, Qk, Vj, Vk

'3', 'Add3', 'yes', 'SUBD', '', '', '1.23', '4.56'

6. Adder 加法器

显示加法器中正在进行的运算。

以下介绍每个字段的含义：

Ins.Id 指令的 id，与指令队列中的相同字段含义相同

Op 操作符

Dst 将要作为哪个保留站的结果输出

Vj 目标寄存器 1 的值

Vk 目标寄存器 2 的值

Progress 本指令的运算进度，格式为：‘已运算周期数/总周期数’

例：

Ins.Id, Op, Dst, Vj, Vk, Progress('5/10')

'4', 'ADDD', 'Add1', '1','2', '1/2'

7. Multiplier　乘法器

显示乘法器中正在进行的运算。

以下介绍每个字段的含义：

Ins.Id 指令的 id，与指令队列中的相同字段含义相同

Op 操作符

Dst 将要作为哪个保留站的结果输出

Vj 目标寄存器 1 的值

Vk 目标寄存器 2 的值

Progress 本指令的运算进度，格式为：‘已运算周期数/总周期数’

例：

Ins.Id, Op, Dst, Vj, Vk, Progress

'5', 'DIVD', 'Mult1','1','2' ,'40/40'

8. LDer 加载器

显示正在进行的 load 指令。

以下介绍每个字段的含义：

Ins.Id 指令的 id，与指令队列中的相同字段含义相同

Op 操作符

Addr 目标地址

Progress 本指令的加载进度，格式为：‘已加载周期数/总周期数’

例：

Ins.Id, Op, Addr, Progress

'6', 'LD', '1234', '1/2'

9. STer 储存器

显示正在进行的 store 指令。

以下介绍每个字段的含义：

Ins.Id 指令的 id，与指令队列中的相同字段含义相同

Op 操作符

Addr 目标地址

FP.Value 将要储存的值

Progress 本指令的储存进度，格式为：‘已储存周期数/总周期数’

例：

Ins.Id, Op, Addr, FP.Value('F0'), Progress

'7', 'SD', '4000', '123', '2/2'

10. CDB 总线

显示总线上目前传输的数据。

以下介绍每个字段的含义：

RS.Name 作为标识符的保留站名，之后的数据将会被发到所有标示着这一保留站名的位置

Value 发送的数据值

例：

RS.Name, Value

'Mult2', '0.0'

11. Memory 内存

显示 Memory Watchlist 中所要求显示的内容。

用户可以通过在 watchlist 中注册内存地址来获得内存的显示列表。

以下介绍每个字段的含义：

Addr 内存中的地址

Value 储存的数据

例：

Addr, Value

'1', '0.0'

12. Timer 计时器

每个周期+1。

==========

==========

函数介绍

1. Load Instructions from File.

可以从文件读取多条指令并加入队列中，包含查错功能。

Load Instructions from String.

可以读入一条指令到队列末尾，包含查错功能。

Clear Instructions.

删除队列中所有指令。

Timer Step n(>=1).

步进，运行 n 个周期，n 默认为 1，若大于所需时间则运行至结束。

Timer to End.

运行至结束。

Global Complete Flag. // COMPLETE = True

全局变量，是否运行结束。

Global Tic Value. // CUR_TIC = 35

全局变量，已运行的周期数，提供给 Timer。

Modify FP(index=[0,10]). GET/SET = '1.23' / True

改变/读取寄存器中的值。

Modify Mem(index=[0,4096)). GET/SET = '4.56' / True

改变/读取内存中的值。