Lab3 实现 Tomasulo 算法模拟器

姓名：李振
学号：SA14225109

实验代码：

Tomasulo.java

可执行文件：

|-----Tomasulo.jar
|-------Tomasulo.bat
在 windows 平台使用鼠标双击 Tomasulo.bat 文件即可运行

设计思想：

程序的 core 函数每次都会依次对指令状态数组逐行遍历，判断每条指令此时的状态；顺序是从左到右，从上至下。

```
If(执行指令没有流出){
    If(满足条件流出){
        -->指令流出
    }Else{
        空
    }
}else if(此时指令已经流出，但没有执行完毕){
    If(若指令没有执行){
        If(满足条件，可以执行){
            -->执行
        }else{
            等待满足条件在执行
        }
    }else{
        //此时指令正在执行中
        ---->修改执行的剩余时间
    }
}else if(如果指令没有写回){
    写回
}
```

缺陷

由于对保留站中的指令判断是根据指令操作符 op 进行判断选择的，因此当保留站中如果同时存在两个相同类型的执行，在执行的时候会出错。其他情况下运行的，都是正确的

对 Tomasulo 的理解与实现

```
boolean instIssue(String op, String rd, String rs, String rt) {
		int IsdoSomething = -1;
		int whichr = -1;
		// 选择空闲的保留站
		if (op.equals("ADD") || op.equals("SUB")) {
			for (int i = 1; i < 4; i++) {
				if (resst[i][2].equals("no")) {
					whichr = i;
					break;
				}
			}
		} else {
			for (int i = 4; i < 6; i++) {
				if (resst[i][2].equals("no")) {
					whichr = i;
					break;
				}
			}
		}
		if (whichr > 0) {
			// 成功找到一个空闲的保留站
			String r = resst[whichr][1];
			for (int i = 1; i < 17; i++) {
				// 检测第一个操作数是否就绪
				if (regst[0][i].equals(rs)) {
					if (regst[1][i].equals("0")) {
						// 第一个操作数就绪，把寄存器 rs 中的操作数取到当前的保
留站 Vj
						resst[whichr][4] = regst[2][i];
						// 置 Qj 为 0，表示当前保留站的 Vj 中的操作数就绪
						resst[whichr][6] = "0";
					} else {
						// 第一个操作数没有就绪
						// 进行寄存器换名，即把将产生该操作数的保留站的编号放入
当前保留站的 Qj
						resst[whichr][6] = regst[1][i];
					}// if-else
				}
				// ----------------------------------------------------
				if (regst[0][i].equals(rt)) {
					if (regst[1][i].equals("0")) {
						// 第二个操作数就绪，把寄存器 rs 中的操作数取到当前的保
留站 Vj
						resst[whichr][5] = regst[2][i];
						// 置 Qk 为 0，表示当前保留站的 Vk 中的操作数就绪
						resst[whichr][7] = "0";
					} else {
						// 第二个操作数没有就绪
						// 进行寄存器换名，即把将产生该操作数的保留站的编号放入
当前保留站的 Qk
						resst[whichr][7] = regst[1][i];
					}// if-else
				}
			}
			resst[whichr][2] = "Yes";
			resst[whichr][3] = op;
			for (int i = 1; i < 17; i++) {
				if (regst[0][i].equals(rd)) {
					regst[1][i] = r;
				}
			}
			IsdoSomething = 1;
			return true;
		} else {
			System.out.println("there is a wrong.");
		}
		if (IsdoSomething > 0) {
			return true;
		} else {
			return false;
```

```
            }
        }
boolean instExcute(String op, String rd, String rs, String rt) {
            // 指令开始执行
            // 这里是要计算执行时间的
            int whichToRun = -1;
            for (int i = 1; i < 6; i++) {
                if (resst[i][3] == op) {
                    whichToRun = i;
                }
            }
            // Time 为空，则判断运算是否符合 进入执行的条件
            if (resst[whichToRun][6] == "0" && resst[whichToRun][7] == "0") {
                // 准备就
                // 判断指令类型，挂上时间
                if (op == "ADD") {
                    resst[whichToRun][0] = Integer.toString(time[0]);
                } else if (op == "SUB") {
                    resst[whichToRun][0] = Integer.toString(time[1]);
                } else if (op == "MULT") {
                    resst[whichToRun][0] = Integer.toString(time[2]);
                } else if (op == "DIV") {
                    resst[whichToRun][0] = Integer.toString(time[3]);
                }// if-else
                return true;
            } else {
                return false;
            }
        }
void instWB(String op, String rd, String rs, String rt) {
            int whichToWB = -1;
            for (int i = 1; i < 6; i++) {
                if (resst[i][3] == op) {
                    whichToWB = i;
                }
            }
            String r = resst[whichToWB][1];
            for (int i = 1; i < 17; i++) {
                // 对于任何一个正在等该结果的浮点寄存器 x
                if (regst[1][i].equals(r)) {
                    // 向该寄存器写入结果
                    if (op == "ADD") {
                        regst[2][i] = resst[whichToWB][4] + "+"
                                + resst[whichToWB][5];
                    } else if (op == "SUB") {
                        regst[2][i] = resst[whichToWB][4] + "-"
                                + resst[whichToWB][5];
                    } else if (op == "MULT") {
                        regst[2][i] = resst[whichToWB][4] + "*"
                                + resst[whichToWB][5];
                    } else if (op == "DIV") {
                        regst[2][i] = resst[whichToWB][4] + "/"
                                + resst[whichToWB][5];
                    }// if-else
                    // 并把该寄存器的状态置为数据就绪
                    regst[1][i] = "0";
                }// ----------------------------------------------
            }
            for (int i = 1; i < 6; i++) {// 对任何一个正在等该结果作为第一个操作数的保留站 x
                if (resst[i][6].equals(r)) {
                    // 向该保留站的 Vj 写入结果
                    if (op == "ADD") {
                        resst[i][4] = resst[whichToWB][4] + "+"
                                + resst[whichToWB][5];
                    } else if (op == "SUB") {
                        resst[i][4] = resst[whichToWB][4] + "-"
                                + resst[whichToWB][5];
                    } else if (op == "MULT") {
                        resst[i][4] = resst[whichToWB][4] + "/"
                                + resst[whichToWB][5];
                    } else if (op == "DIV") {
                        resst[i][4] = resst[whichToWB][4] + "*"
                                + resst[whichToWB][5];
                    }// if-else
                    // 置 Qj 为 0，表示该保留站的 Vj 中操作数就绪
                    resst[i][6] = "0";
```

```
                }
            }

            for (int i = 1; i < 6; i++) {// 对任何一个正在等该结果作为第二个操作数的保留站 x
                if (resst[i][7].equals(r)) {
                    // 向该保留站的 Vk 写入结果
                    if (op == "ADD") {
                        resst[i][5] = resst[whichToWB][4] + "+"
                                + resst[whichToWB][5];
                    } else if (op == "SUB") {
                        resst[i][5] = resst[whichToWB][4] + "-"
                                + resst[whichToWB][5];
                    } else if (op == "MULT") {
                        resst[i][5] = resst[whichToWB][4] + "/"
                                + resst[whichToWB][5];
                    } else if (op == "DIV") {
                        resst[i][5] = resst[whichToWB][4] + "*"
                                + resst[whichToWB][5];
                    }// if-else

                    // 置 Qk 为 0，表示该保留站的 Vk 中操作数就绪
                    resst[i][7] = "0";
                }
            }
            // 释放当前的保留站，将它置为空闲状态
            resst[whichToWB][0] = "";
            // resst[whichToWB][1] = "?";
            resst[whichToWB][2] = "no";
            resst[whichToWB][3] = "";
            resst[whichToWB][4] = "";
            resst[whichToWB][5] = "";
            resst[whichToWB][6] = "";
            resst[whichToWB][7] = "";
    }
```

上述代码只对浮点运算指令进行流入--执行--写回，对 load/store 指令是在 core 函数中判断分析的。

实验分析结论

1. 模拟器完成后，用你的模拟器测试以下程序并答题。

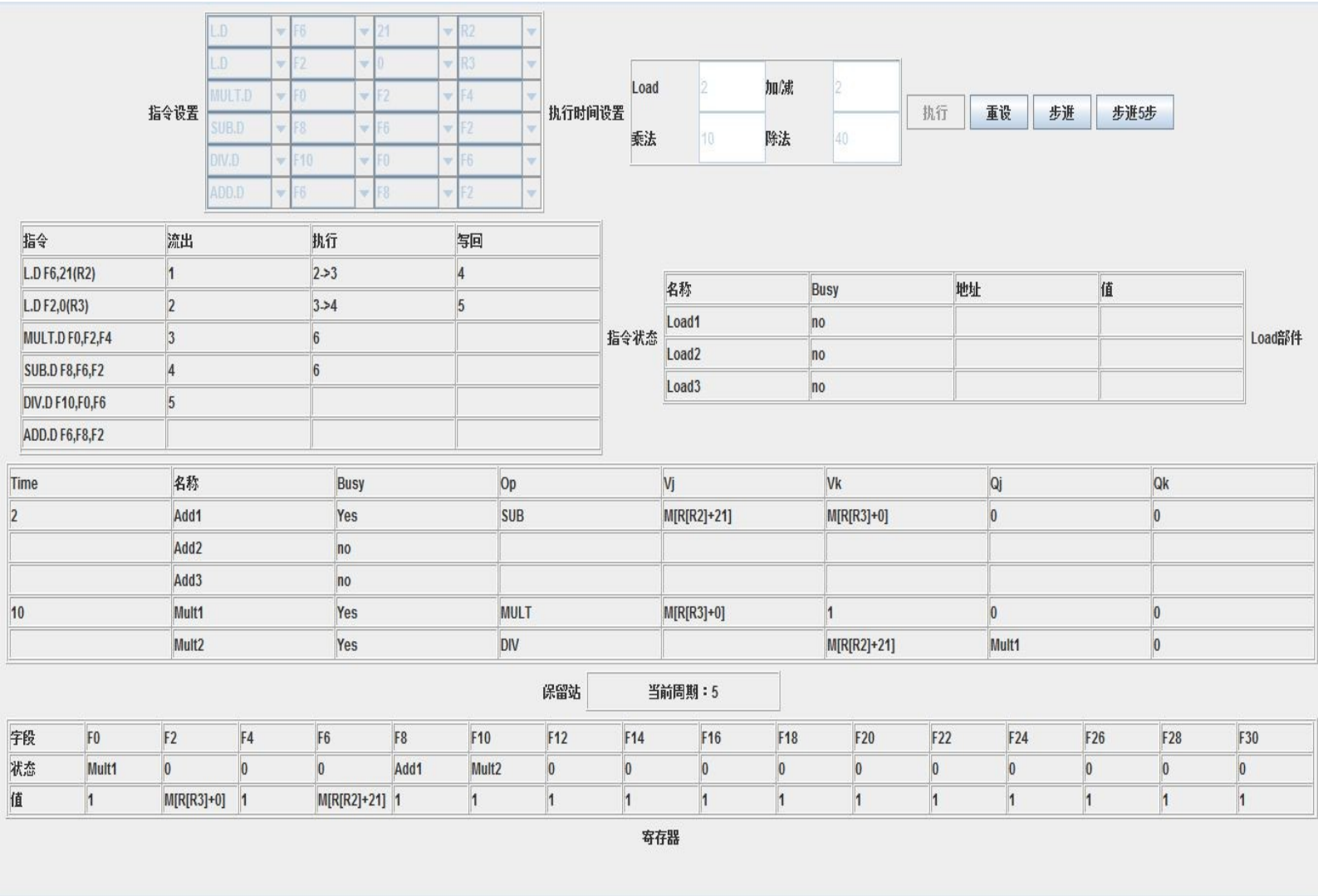
```
L.D   F6, 21（R2）
L.D   F2, 0（R3）
MUL.D    F0, F2, F4
SUB.D    F8, F6, F2
DIV.D    F10, F0, F6
ADD.D    F6, F8, F2
```

说明：

（1）假设浮点功能部件的延迟时间：加减法 2 个周期，乘法 10 个周期，load/store2 个周期，除法 40 个周期。而指令的流入和写回为 1 个周期。

（2）第一条指令中 21(R2)表示一个内存地址，将该地址的值 load 到 F6 寄存器。该内存的内容可以事先指定

a. 给出在第 5 个时钟周期后，保留站的内容。



指令	流出	执行	写回
L.D F6,21(R2)	1	2->3	4
L.D F2,0(R3)	2	3->4	5
MULT.D F0,F2,F4	3	6	
SUB.D F8,F6,F2	4	6	
DIV.D F10,F0,F6	5		
ADD.D F6,F8,F2			

指令状态

名称	Busy	地址	值
Load1	no		
Load2	no		
Load3	no		

Load部件

Time	名称	Busy	Op	Vj	Vk	Qj	Qk
2	Add1	Yes	SUB	M[R[R2]+21]	M[R[R3]+0]	0	0
	Add2	no					
	Add3	no					
10	Mult1	Yes	MULT	M[R[R3]+0]	1	0	0
	Mult2	Yes	DIV		M[R[R2]+21]	Mult1	0

保留站 当前周期：5

字段	F0	F2	F4	F6	F8	F10	F12	F14	F16	F18	F20	F22	F24	F26	F28	F30
状态	Mult1	0	0	0	Add1	Mult2	0	0	0	0	0	0	0	0	0	0
值	1	M[R[R3]+0]	1	M[R[R2]+21]	1	1	1	1	1	1	1	1	1	1	1	1

寄存器

注意：寄存器第三行中值 1 是初始值。

b. 给出在第 10 个周期后，保留站，寄存器状态表的信息。

指令设置

L.D	F6	21	R2
L.D	F2	0	R3
MULT.D	F0	F2	F4
SUB.D	F8	F6	F2
DIV.D	F10	F0	F6
ADD.D	F6	F8	F2

执行时间设置

Load	2	加/减	2
乘法	10	除法	40

执行 重设 步进 步进5步

指令	流出	执行	写回
L.D F6,21(R2)	1	2->3	4
L.D F2,0(R3)	2	3->4	5
MULT.D F0,F2,F4	3	6	
SUB.D F8,F6,F2	4	6->7	8
DIV.D F10,F0,F6	5		
ADD.D F6,F8,F2	6	9->10	

指令状态

名称	Busy	地址	值
Load1	no		
Load2	no		
Load3	no		

Load部件

Time	名称	Busy	Op	Vj	Vk	Qj	Qk
	Add1	no					
0	Add2	Yes	ADD	M[R[R2]+21]-M[R[R3]+0]	M[R[R3]+0]	0	0
	Add3	no					
5	Mult1	Yes	MULT	M[R[R3]+0]	1	0	0
	Mult2	Yes	DIV		M[R[R2]+21]	Mult1	0

保留站 当前周期：10

字段	F0	F2	F4	F6	F8	F10	F12	F14	F16	F18	F20	F22	F24	F26	F28	F30
状态	Mult1	0	0	Add2	0	Mult2	0	0	0	0	0	0	0	0	0	0
值	1	M[R[R3]+0]	1	M[R[R2]+21]	M[R[R2]+21...	1	1	1	1	1	1	1	1	1	1	1

寄存器