Tomasulo 算法模拟器设计与分析实验说明

（一）实验简介

本实验要求同学们使用 C/C++，python，Java 等常见编程语言设计一个用户态的 Tomasulo 动态流水线调度算法模拟器，能够接受一种被称为 NEL（相关定义参见附录）的最简单汇编语言作为输入，模拟它在特定硬件平台（相关定义参见附录）的执行情况（各类指令的假设运行时间在附录中给出），给出任意时钟周期的指令状态，保留栈状态，寄存器结果状态和寄存器数值等目标结果。

（二）实验要求：

整个实验总分 100 分，其中基础实验内容占 80 分，在基础分数上完成扩展实验内容可获得 5-60 分的加分，该加分除了可以算作本次实验的得分之外，还可以用于弥补第一次实验丢失的分数，**两次实验的总分数合计不超过 200 分，在总评中占 22%**。

基础实验要求：

要求你设计的模拟器需要：

1、能够**正确接受任意 NEL 汇编语言编写的指令序列作为输入**（1000 行以内，保证符合附录中定义的文法）

2、能够正确输出**每一条指令发射的时间周期，执行完成的时间周期，写回结果的时间周期**。

3、能够正确输出**各时间周期的寄存器数值**。

4、能够正确输出**各时间周期保留站状态、LoadBuffer 状态和寄存器结果状态**。

扩展实验要求（选择完成下列内容中若干项，根据完成质量和完成度评分）：

1、**设计美观的交互界面**。

2、**实现高效的模拟算法**，能够支持更大规模的 NEL 指令序列。

3、**调研不同的分支预测技术**，实现或改进部分技术，设计自己的分支预测方案等。

4、**丰富 NEL 语言**，为它添加更多的指令支持，并能够模拟这些指令的执行。

（四）实验检查要求：

请在 6 月 5 日之前在网络学堂提交代码和实验报告的 ZIP 格式压缩包文件，文件命名为 学号_姓名.zip 的形式（例如 2016011000_李四.zip），请在实验报告中说明你的模拟器设计思路，以及你所完成的功能，并请说明你代码的编译运行的方式，此外**请自行准备一些 NEL 语言编写的样例小程序，并在实验报告中说明这些程序的预期结果和实际模拟结果是否一致以证明你实现的模拟器的正确性**。

提交代码之后需要给助教当面演示模拟器的运行效果，时间地点另行通知，可详细介绍你所做的功能扩展（如果有的话），此外**助教还会在现场再额外提供一些测试样例用于测试你程序的正确性**（保证这些测试样例中的指令均符合附录中的文法定义，**你需要给出在不做任何分支预测的情况下某些时间周期各类目标结果的数值**）。

（五）其他说明：

1、因为存在循环，同一条指令可能会被多次执行，你只需要**输出第一次执行该指令时**的发射周期，执行完成周期和写回结果周期。

2、由于 NEL 语言没有内存模型，因此在 Tomasulo 算法中**所有涉及内存地址和访存结果的位置都使用十六进制立即数表示**即可。

3、可以参考开源模拟器的实现方案，也可以使用一些开源框架，但请在引用文献中说明。如果开源框架功能过于强大可能会影响到相应部分的实验评分。

附录 1：NEL 汇编语言简介

NEL 是我们自定义的简单汇编语言，其文法定义如下：

1. 词法定义（正则表达式）

DIGIT = ([0-9])

HEX_DIGIT=([0-9A-F])

DEC_INTEGER=({DIGIT}+)

HEX_INTEGER=(0x{HEX_DIGIT}+)

INTEGER={HEX_INTEGER}

REGISTER=(F{DEC_INTEGER})

关键字：

“ADD”，“MUL”，“SUB”，“DIV”，“LD”，“JUMP”

2.语法定义（CFG 语法定义）

Program := InstList

InstList := Inst

InstList := Inst ‘\n’ InstList

Inst := OPR ‘,’ REGISTER ‘,’ REGISTER ‘,’ REGISTER

Inst := “LD” ‘,’ REGISTER ‘,’ INTEGER

Inst := “JUMP” ‘,’ INTEGER ‘,’ REGISTER ‘,’ INTEGER

OPR := “ADD” | “MUL” | “SUB” | “DIV”

3.说明

1、NEL 中的整数统一按照带符号 32 位补码的形式存储，表示一律使用十六进制表示（0x 开头，字母位大写）

2、NEL 是一种由连续三地址码指令组成的汇编语言，它只支持三类指令，分别是运算指令（ADD，SUB，MUL，DIV）、装载指令（LD）和控制指令（Jump）。它没有内存模型，所有的运算都首先通过使用 Load 将数字读入寄存器，经过寄存器之间的各类运算获得结果，此外还可以通过 Jump 来实现控制。

3、基本运算指令语义：NEL 的运算指令都可以表示为 OPR1 ‘,’ REGISTER ‘,’ REGISTER ‘,’ REGISTER（例如 ADD,F1,F2,F3）的形式，上述例子的语义为针对 F2 和 F3 寄存器中的数值进行 ADD 运算，将结果存入 F1 中，其他指令的含义以此类推，（**除法指令均表示整除，特别的，如果除数位置寄存器数值为零，该语句无效**）

4、装载指令语义：NEL 的装载指令都可以表示为“LD” ‘,’ REGISTER ‘,’ INTEGER 的形式 (如 LD,F1,0xFFFF)，上述例子的语义为将 0x0000FFFF 装载到 F1 中。

5、跳转指令语义：NEL 的跳转指令都可以表示为“JUMP” ‘,’ INTEGER ‘,’ REGISTER ‘,’ INTEGER 的形式（如 **JUMP,0,F1,0xFFFFFFFD**），上述例子的语义为如果 F1 寄存器中的数值为 0 的话（严格等于零），那么将向前从该指令开始向前数 3 句执行（**比如当前指令编号为 3，那么将从编号为 0 的指令开始执行**）

6、寄存器栈模型：NEL 支持无限个寄存器，每一个寄存器都是 32 位的，寄存器编号按十进制编码，分别为 F0,F1,F2....，但是在测试用例中只需要考虑 32 个寄存器即可（F0-F31），默认所有寄存器的初始数值均为 0。

4.样例

下列便是一个简单的 NEL 语言的例子

```
LD,F1,0xC
LD,F2,0xFFFF3C6F
LD,F3,0xFFFFFFFA
LD,F4,0x0
ADD,F2,F4,F2
DIV,F4,F2,F3
MUL,F4,F3,F3
LD,F5,0xFFFFFF32
LD,F18,0x1
SUB,F5,F4,F2
SUB,F1,F1,F18
JUMP,0x0,F1,0xFFFFFFF9
```

附录 2：需要模拟的硬件平台：

我们假设有一个硬件平台包含如下功能部件

3 个加减法器（Add1, Add2, Add3）

2 个乘除法器（Mult1, Mult2）

2 个 Load 部件（Load）

同时我们假设有如下保留站：

6 个加减法保留站（Ars1, Ars2, ...）

3 个乘除法保留站（Mrs1, Mrs2, ...）

3 个 LoadBuffer（LB1, LB2, ...）

注意：

在本硬件平台中，保留站的数量要多于功能部件的数量，因此保留站中就绪的指令将构成一个先进先出队列，每当功能部件出现空闲时，取最先就绪的指令进入功能部件执行，如果有两条指令同时就绪，那么就按照它们编号（即是指令序列中的第几条指令）大小排序，编号比较小的指令默认排在更靠前的位置。

附录 3：各类指令的假定运行时间：

LD 指令需要 3 个周期完成；JUMP 指令需要 1 个周期完成；ADD 和 SUB 指令需要 3 个周期完成；MUL 指令需要 12 个周期完成；DIV 指令需要 40 个周期完成