# Gardenia

## ——计算机系统大作业报告

F1124004 5110309058 陈志鹏

F1124004 5110309028 陈楠昕

目录

[一．简介 2](#_Toc359694569)

[二．ISA设计 2](#_Toc359694570)

[三．Compiler设计 3](#_Toc359694571)

[四．CPU设计 3](#_Toc359694572)

[五．Cache设计 5](#_Toc359694573)

[六．组内分工 5](#_Toc359694574)

[七．使用介绍 5](#_Toc359694575)

[八．测试结果 6](#_Toc359694576)

[九．总结与致谢 6](#_Toc359694577)

# 一．简介

这个项目是一个CPU及相应编译器、汇编器的设计，由陈志鹏和陈楠昕合作完成。CPU是Harvard结构的Tomasulo设计，含64个寄存器，指令集为32位RISC，并且实现了Multiple Issue及Cache，用verilog描述；编译器和汇编器用Java实现。

# 二．ISA设计

ISA参考MIPS设计，但基于项目需要有适当修改。每条指令为32位，前4位为指令代码。

对于部分指令，最后一位表示该指令是否包含立即数，最后一位为0表示最后一个参数是寄存器，如果为1说明最后一个参数是立即数（Imm）。由于有64个寄存器，需6个二进制位表示一个寄存器。

对于bgt和bnez，所要跳转的地址在汇编代码中用label表示，在二进制代码中用基于该指令的偏移量表示。

以下为汇编指令和对应二进制码，其中下标表示所占位数：

|  |  |
| --- | --- |
| add | |
| 汇编 | add dst, src1, src2/Imm |
| 二进制（无立即数） | 1000 dst6 src16 src26 00 00000000 |
| 二进制（有立即数） | 1000 dst6 src16 Imm15 1 |
| mul | |
| 汇编 | mul dst, src1, src2/Imm |
| 二进制（无立即数） | 1001 dst6 src16 src26 00 00000000 |
| 二进制（有立即数） | 1001 dst6 src16 Imm15 1 |
| bgt | |
| 汇编 | bgt src1, src2, label |
| 二进制 | 1010 src16 src26 offset16 |
| bnez | |
| 汇编 | bnez src1, label |
| 二进制 | 1011 src16 offset22 |
| lw | |
| 汇编 | lw dst, src1, src2/Imm |
| 二进制（无立即数） | 1100 dst6 src16 src26 00 00000000 |
| 二进制（有立即数） | 1100 dst6 src16 Imm15 1 |
| sw | |
| 汇编 | sw dst, src1, src2/Imm |
| 二进制（无立即数） | 1101 dst6 src16 src26 00 00000000 |
| 二进制（有立即数） | 1101 dst6 src16 Imm15 1 |
| j | |
| 汇编 | j label |
| 二进制 | 1110 offset28 |
| mv | |
| 汇编 | mv dst, src1/Imm |
| 二进制（无立即数） | 1111 dst6 src16 00000000 00000000 |
| 二进制（有立即数） | 1111 dst6 Imm21 1 |
| nop | |
| 汇编 | nop |
| 二进制 | 00000000 00000000 00000000 00000000 |
| halt | |
| 汇编 | halt |
| 二进制 | 00010000 00000000 00000000 00000000 |

# 三．Compiler设计

Compiler设计是用Java描述的，支持的语法仅为样例程序中所用到的，仅支持将样例程序更改常数或者用完全相同的句法。Compiler的词法分析使用JFlex完成，语法分析使用JCup完成，构建了AST然后转成汇编代码。

针对矩阵乘法，Compiler采用了Loop Unrolling技术，将最内层循环完全展开。一开始打算一次最多issue 26条指令，Compiler根据最内层循环体部分重新排列代码并分块，在中间插入nop指令表示一次issue的结束。后面改为一直issue直到需要stall。

# 四．CPU设计

CPU设计为Tomasulo，但将EX和WB两个阶段合为一个，即一旦指令完成就立刻把结果送向所需的运算单元。

CPU包含以下几个文件：

|  |  |
| --- | --- |
| CPU.v | 整个CPU最高层模块 |
| clock.v | 以2ps为周期提供方波输出 |
| fetch.v | 从instcache获取指令，解码后放入Reservation Station |
| RS.v | Reservation Station、Register File |
| ALU.v | 加法器、乘法器 |
| instcache.v | 指令缓存 |
| datacache.v | 数据缓存 |
| instmem.v | 指令内存，从input.bin中读取程序；到内存取指令有100周期延迟 |
| datamem.v | 数据内存，从ram\_data.txt中读取数据并将最后结果放入cpures.txt；内存读写有100周期延迟 |
| define.v | 定义timescale和程序中所用到的常量 |

CPU结构如下图所示：

1. CPU：

包含clock和fetch，将clock产生的时钟波形传给fetch。

1. fetch：

通过Instruction Cache获取指令，每次获取多条指令，解码后传给Reservation Station，若Reservation Station满或遇到bgt指令操作数没有得到或Instruction Cache Miss则stall。

1. Reservation Station：

将fetch传来的解码后指令放入表中。CPU中共有96个lw单元，32个add单元，32个mul单元，32个sw单元，需要8个二进制位标记运算单元。表的每一项设计如下：

add、mul、lw：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Busy1 | Value132 | Value232 | Quest18 | Quest28 | Valid11 | Valid21 |

Busy表示该运算单元是否被在占用，Valid1和Valid2表示该运算单元的两个操作数是否可用。Quest1和Quest2表示操作数来自哪个运算单元，可由Register Status获知，若操作数是立即数或从寄存器中去出，则Quest无效。可用的操作数的值保存在Value1和Value2里面。

sw：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Busy1 | Value132 | Value232 | Value332 | Quest18 | Quest28 | Quest38 | Valid11 | Valid21 | Valid31 |

各二进制位意义同上。

在各个操作数都可用时，该运算单元就会执行，得出结果后，把结果送给需要的运算单元或寄存器，最后清零。

1. Register File：

每个寄存器有一个8位的Register Status，表示要写入该寄存器的运算单元。若没有运算单元要写入该寄存器，则Register Status为01111111，表示寄存器的值有效。

1. Memory Access Queue：

这个元件目的是防止lw和sw乱序执行，该元件实质是一个循环队列，会依次按照指令issue进入的顺序处理每个lw和sw。

# 五．Cache设计

Cache共8个Block，每个Block大小为128Byte，用Direct-Map方式。

instcache是只读的，datacache可读可写，用write back方式，如果write miss，将dirty 的block写入内存再把新block拿到cache并写入数据。

Cache每次miss时从Memory拿两个相邻的block回来，instcache返回从请求地址开始的一个block。

在最后一条指令halt被issue进入Reservation Station时，Reservation Station发出一个信号，通知datacache将所有dirty的block写回datamem中，然后通知datamem将数据写入文件。

# 六．组内分工

陈志鹏：ISA的设计，CPU的总体设计，编写汇编器，设计并编写RS、instmem、datamem、datacache和ALU，参与最后整合与调试，起草并修改报告

陈楠昕：设计并编写编译器、fetch和instcache，最后整合并调试，调整CPU参数，修改报告。

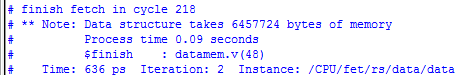
# 七．使用介绍

编译器和汇编器：编译运行decency.compiler.Main或运行jar文件(不需要加参数)，输入文件为test.c，输出文件为test.s和test.bin，其中test.s为汇编代码，test.bin为二进制代码。

CPU：使用modelsim仿真bourgeois中的CPU.v，输入文件为二进制代码input.bin和内存原数据ram\_data.txt，输出文件为cpures.txt，为CPU仿真结束后内存数据。

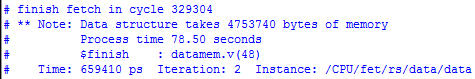
# 八．测试结果

对于给出的样例数据的运行时间(其中finish fetch指的是最后一条指令halt issue的时间)：



即318个cycle。

对于自己生成的30\*30大小的矩阵乘法运行时间：



即329705个cycle。

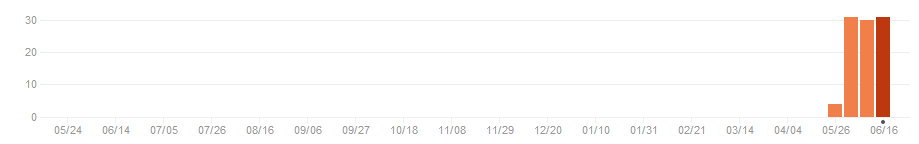
生成代码可见makedata.cpp。

测试得出来的结论是运行速度很大程序上取决于cache，调整运算单元个数很大程度上也能提高性能，因为瓶颈部分应该在于Memory Access Queue，而这里的并行性很差，因此适当减少lw和sw单元增加其他单元应该能有效提高效率。但由于时间关系没有进行太多的尝试。

# 九．总结与致谢

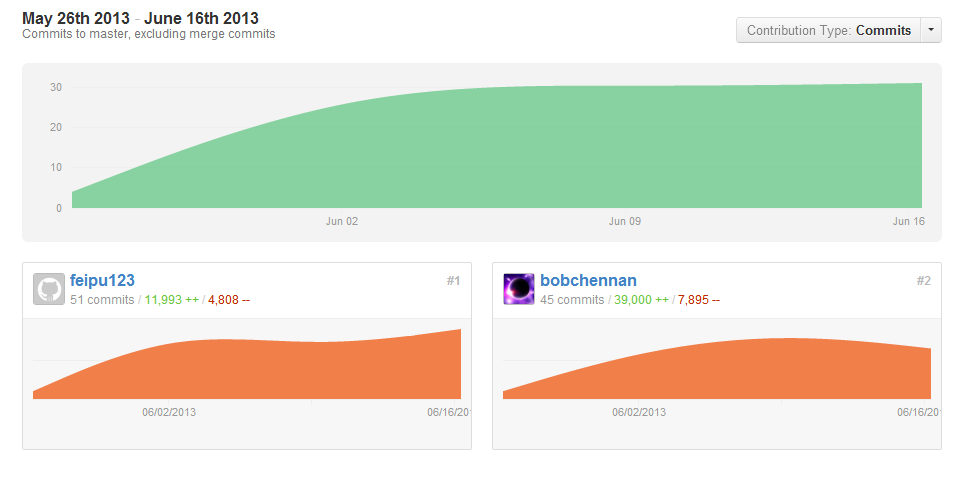
在这个项目中，陈楠昕与陈志鹏并肩协力，共克千难万险，成功完成该项目。下面是Github上面的一些数据统计。

Commit activity

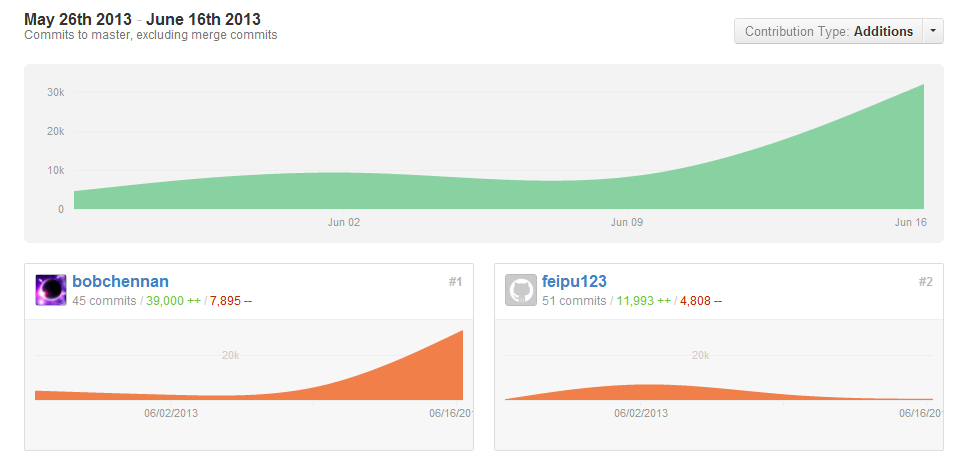




Commits（左为陈志鹏，右为陈楠昕）：



Additions（左为陈楠昕，右为陈志鹏）：



Language statics：



同时，非常感谢李青林提供的帮助。