计算机组成大作业报告

Report of MS108 project

李一同
5110309044
ACM11 F1124004, SJTU
lrank@sjtu.edu.cn

吴航
5110379024
ACM11 F1124004, SJTU
abcwuhang@sjtu.edu.cn

Abstract

MS108计算机系统的大作业（project）要求我们设计一个CPU，可以实现矩阵乘法运算。

我们设计的CPU为经典五级pipeline，指令集长度为32位，支持一些简单的整型运算，并不支持浮点运算。

Keywords CPU-simulator, pipeline

1. Introduction

这是一份关于计算机组成（MS108）大作业的报告，作于2013 年Spring Semester。由我（李一同）和吴航同学两人共同完成。

1.1 Cooperation and Division

本次大作业大致分为前端和后端两个部分。

前端部分包括编译器与汇编器，后端包括流水线作业。李一同同学负责处理后端中pipepline的编写，吴航同学负责处理前端，memory和cache的设计以及前后端的接口。

1.2 Environment

- Compiler是用JAVA语言编写，环境为Eclipse Juno，JDK1.7
- Simulator是用verilog编写，测试环境为ModelSim SE-64 10.1c。使用Icarus Verilog （iverilog）辅助测试。

2. 编译器与汇编器

编译器负责编译源程序并生成仿mips代码，汇编器负责把代码翻译成机器码。编译器采用了许多优化算法以减少重复操作及停顿，如公共子表达式消除、强度削减、循环展开等。具体可参见用户手册。我设计的指令有以下几种：load、store、li （把立即数载入寄存器）、addu、addiu、sll（左移位运算）、mul （乘法）、bge （分支指令）、j、muli（带立即数的乘法），另外还有两种特殊指令：空指令及停止指令（用来指示流水线何时停止）。汇编器进一步生成32位等长机器码，格式为：前4位为操作符码，紧接其后的是5 位寄存器号码或立即数。

如Table.1：

Table 1. 指令表

指令	仿mips代码	翻译机器码（格式）
Load	Lw $rd,imm($rs)	0000(5’ $rd)(5’ $rs)(18’ imm)
Store	Sw $rs,imm($rt)	0001(5’ $rs)(5’ $rt)(18’ imm)
Li	Li $rd,imm	0010(5’ $rd)(23’ imm)
Addu	Addu $rd,$r1,$r2	0011(5’ $rd)(5’ $r1)(5’ $r2)(13’ b0)
Addiu	Addiu $rd,$r1,imm	0100(5’ $rd)(5’ $r1)(18’ imm)
Sll	Sll $rd,$r1,imm	0101(5’ $rd)(5’ $r1)(18’ imm)
mul	Mul $rd,$rs,$rt	0110(5’ $rd)(5’ $rs)(5’ $rt)(13’ b0)
Bge	Bge $rs,$rt,address	0111(5’ $rs)(5’ $rt)(18’ address)
J	J address	1000(28’ address)
Muli	Mul $rd,$rs,imm	1001(5’ $rd)(5’ $rs)(18’ imm)
nop	\	11100…0
stop	\	111..11（32位）

3. 流水线

流水线的设计是参照经典的5级pipeline（带hazard detect 以及bypassing），分别为IF，ID，EX，MEM，WB五个部分。

3.1 Pipeline

每个部分的基本功能及设计思路如下：

- IF：
  进行PC运算，以及instruction fetch的工作
- ID：
  进行instruction decode工作，并去寄存器堆中取出对应的操作数（如果有的话），确定下一级的操作符。
- EX：
  对ID阶段取出的操作数进行运算。
- MEM：
  内存读取工作（如果有的话）。
- WB：
  将操作完的目标数，存会目标寄存器。

我用四个元件IF/ID，ID/EX，EX/MEM，MEM/WB来控制时钟，将这五个流水级来分级。

3.2 Hazard Detecting

冒险侦测主要侦测两种情况的冒险：

- 跳转语句：
  在我们的结构中，跳转语句有两种，分别是J和BGE，对这两个指令操作的基本思路是一样的。通过EX后得到需要的地址，将它传给IF中的PC 单元实现PC的跳转。

我们并没有实现branch predict，所以在得到PC 的目标地址前，总是提供两个Bubble。

- RAW Hazard： 这是一个经典的Hazard，在五级pipeline中并没有好办法能将它消去。

  我的解决方案是在R和W直接插入Bubble，由于我做了Bypassing （Forwarding），这样只需要一个Bubble即可。

3.2.1 Analysis

两种Hazard是拖CPI后腿的主要问题，虽然前者需要两个Bubble，但是后者在矩阵乘法代码中出现的次数明显要更多（而且我们在编译器中做了loop unrolling 的优化，会大大减少跳转语句的使用），所以我们并没有实现branch predict。主要关注后者Bypassing和Forwarding这方面的优化。

3.3 Bypassing

我直接将MEM的结果回传IF。

可以再加一个将EX的结果回传到IF的操作，这样做的好处是可以省去一些指令（如ADD，MUL）的那一个Bubble。

但我并没有这么写，主要是为了统一LW和其他指令在结构上的一致性。

4. 仿真结果

我们对所给的sample进行了仿真，clock数在315，可见效果还是不错的。

另外，我们自行测试了一组30*30级别的矩阵相乘的数据，clock数大约在54,000。

最后，我们通过编程，验证了程序的正确性。