清华大学计算机科学与技术系

MIPS-lite Simulator

《计算机系统结构》课程实验逻辑综述

kde9

kfirst 孔祥欣[1]

deepsolo 胡玮玮[2]

edwardtoday 卿培[3]

2009/6/2

[1] 2006011299, CS62, Mobile: 13401086576, Email: kxx006@gmail.com

[2] 2006011293, CS62, Mobile: 13810313760, Email: huww06@gmail.com

[3] 2006011291, CS62, Mobile: 15901033612, Email: edwardtoday@gmail.com

目录

逻辑分析

组件组成：

经过对流水线的分析，我们的 CPU 的组成主要包括以下部分：

PCUnit、IF2IDReg、ID2EXEReg、EXE2MEMReg、MEM2WBReg、Control、MemInterface、stop2period、ChoRegWVal、RegHeap、Transfer、ChoALU1、ChoALU2、ALU、TransferChoPCVal、ChoPCCtrl

同步实现机制：

由于 java 语言没有像 VHDL 语言那样的语句并行执行的特性，因此若想实现如同 VHDL 语言一样的周期执行的特性，就必须处理好同步的问题。

我们实现周期执行的方式如下：

首先建立 2 个信号量池，用于保存周期开始时各个信号量的值和周期结束时各个信号量的值。在周期中，各个组件按照相互的依赖关系依次执行，并将输出写入到周期结束信号量池中， 以供以后执行的组件使用。

各个组件的运行次序如下：

PCUnit → IF2IDReg → ID2EXEReg → EXE2MEMReg → MEM2WBReg → Control → MemInterface → stop2period → ChoRegWVal → RegHeap → Transfer → ChoALU1 → ChoALU2 → ALU → TransferChoPCVal → ChoPCCtrl

次序在前的组件先执行，次序在后的组件后执行，每个周期所有组件执行一遍。

数据相关分析

典型数据相关(无 LW 指令)

例如： ADD R1 R2 R3 (R3←R1+R2)

SUB R3 R4 R5 (R5←R3-R4)

AND R3 R1 (R3←R3 or R1)

即在上一条是修改某寄存器值，之后一条或之后两条就要用到该寄存器的新值，由于是流水线，取寄存器值时上一条指令还没有进行到写回阶段，取的值是旧的，造成数据相关。

解决方法：旁路技术

在 EXE 阶段加入转发单元。将每条指令要用的寄存器号连同是否可能发生数据冲突的标志位传入转发单元，判断 ID_EXE 的锁存器中要用的有效的寄存器号是否跟 EXE_MEM 锁存器和 MEM_WB 锁存器中的写回寄存器号相同，如果相同就需要将前一条指令刚经过 EXE 阶段的运算结果(尚在 MEM 阶段)，或前两条指令刚经过 MEM 阶段(尚在 WB 阶段)的写回结果传给本条指令 ALU 的源操作数(需要给多路选择器一个操作码选择本旁路)。如此，流水线不需要停止，典型的数据相关问题就可以解决了。

带有 LW 指令的数据相关

例如： LW R1 R2 0 (R2←Mem[R1+0])

ADD R1 R2 R3 (R3←R1+R2)

即上一条是 lw 取内存数据到某寄存器的指令，之后一条就要用到该寄存器的值。由于 lw 需要经过 MEM 阶段才能给出，所以上面解决典型数据相关的旁路技术就无法解决这中数据相关。此时，流水线至少要停止一个周期，除此之外还要添加针对性的控制命令。这样实现起来比较复杂，而我们的解决方法是简单考虑，直接在 ID 阶段判断是否为 lw 指令，如果是的话，直接发出让流水线停止两个周期，这样一来，要用的寄存器值在 EXE 阶段前就会前一条 lw 指令就已经到达写回阶段了。如此，带有 lw 指令的数据相关问题就可以解决了。

结构相关分析：

由于指令与数据存储在同一个内存中，IF 阶段和 MEM 阶段的内存读写不可能同时进行。对于这个问题，我们采用了经典的解决方法，即，对于可能引起结构冲突的 lw、sw 指令，由 control 单元给出标志信号，并向下传递。当 WB 阶段检测到结构冲突时，优先进行数据读写，并将指令读写过程暂停一个时钟周期，即可解决结构冲突。

模块简介

CPU 部分

PCUNIT (ORG.KDE9.PCUNIT/PCUNIT.JAVA)

输入信号量：

类型	名称	作用
int	lastPc	上一个 pc 值或将要跳转的 pc 值
boolean	reset	重置标志位
boolean	hold	保持标志位

输出信号量

类型	名称	作用
int	pc	下一个 pc 值

主要功能：读取 lastPC 的值，若此时 hold 为 0 则将其加 1 后赋给 pc，否则 pc 不变。当遇到 reset 信号时 pc 置为 0。

MEMINTERFACE (ORG.KDE9.MEMORY/MEMINTERFACE.JAVA)

输入信号量：

类型	名称	作用

int	pc	要读取的指令地址
int	memAddr	要读取的数据地址
boolean	we	内存写使能
int	memWVal	内存写入值
boolean	Islwsw	是否为 lw、sw 指令

输出信号量

类型	名称	作用
int	ins	从内存中读出的值
boolean	ready	读出的值是否有效

主要功能：整合 Cache 和 Memory，为 CPU 访存提供统一接口。通过 islwsw 标志位判断当前是指令读取还是数据读取。若为指令读取，则读取指令 cache，若指令 cache 缺失则读取内存。若为数据读取，则读取数据 cache，若数据 cache 缺失则读取内存。Cache 缺失时，置标志位 ready 为 false，则 CPU 将等待一个周期继续尝试。

IF2IDREG (ORG.KDE9.REGISTER/IF2IDREG.JAVA)

输入信号量：

类型	名称	作用
int	InsIn	需要暂存的数据或指令
int	PCIn	需要暂存的 PC 信号
boolean	reset	同步复位，为 1 时复位

输出信号量

类型	名称	作用
int	InsOut	输出到下一阶段的数据或指令
int	PCOut	输出到下一阶段的 PC 信号

主要功能：IF 与 ID 阶段之间的寄存器，用于在周期传递信号量值。

STOP2PERIOD (ORG.KDE9.OTHERS/STOP2PERIOD.JAVA)

输入信号量：

类型	名称	作用
boolean	needStop	是否存在结构冲突
boolean	islwsw	是否存在 lwsw 数据冲突
boolean	ready	内存读出值是否有效

输出信号量

类型	名称	作用
boolean	hold	PCUnit 传递的 hold 信号
boolean	holdx	PCUnit 传递的 holdx 信号

主要功能：通过当前的 CPU 状态，判断流水线是否需要暂停。当 ready 为 false 时，表示 Cache 发生缺失，应暂停一个周期。当 needStop 为 true 时，表示当前指令为 lw 或 sw，应暂停二个周期，当 islwsw 为 true 时表示出现结构冲突，应暂停一个周期。

CONTROL (ORG.KDE9.CONTROL/CONTROL.JAVA)

输入信号量：

类型	名称	作用
int	Ins	输入的指令编码

输出信号量

类型	名称	作用
int	RegAddr1	解析出的第一个寄存器地址
boolean	RegAddrE1	第一个寄存器是否有效

int	RegAddr2	解析出的第二个寄存器地址
boolean	RegAddrE2	第二个寄存器是否有效
int	RegWAddr	解析出的要写入的寄存器地址
int	Im	解析出的立即数
boolean	CChoALU2	ALU 第二输入选择器控制码
int	CChoRegWVal	寄存器写回选择器控制码
int	CChoPCCtrl	PC 数据选择器控制码
int	CALU	ALU 控制码
boolean	CRegWE	寄存器堆写使能
boolean	CMemWE	内存写使能
boolean	storePC	是否保存 PC 值
boolean	needStop	是否存在结构冲突
boolean	islwsw	是否为 lwsw 指令

主要功能：通过分析输入的指令码，提供除转发单元以外的各个过程的控制信号以及下一阶段需要的数据。

REGHEAP (ORG.KDE9.REGISTER/REGISTERHEAP.JAVA)

输入信号量：

类型	名称	作用
int	RegAddr1	第一个寄存器的地址
int	RegAddr2	第二个寄存器的地址
boolean	WE	寄存器写使能
int	RegWAddr	寄存器的写地址
int	RegWVal	要写入的值
boolean	storePC	是否需要 SP 寄存器保存 PC 值
int	PC	正在 ID 阶段的指令地址

boolean	reset	复位信号

输出信号量

类型	名称	作用
int	RegVal1	第一个寄存器的值
int	RegVal2	第二个寄存器的值

主要功能：寄存器堆，通过给出寄存器编号给出寄存器的值，同时在此模块内部，通过判断要写入的寄存器编号与要读出的寄存器编号的关系，处理同时读写而产生的数据冲突。

寄存器堆中共设 32 个寄存器，其中 0001 – 1110 可为用户使用，0000 寄存器为 ZERO 寄存器，值恒零；1111 为 RA 寄存器，保存跳转指令的返回地址。

ID2EXEREG (ORG.KDE9.REGISTER/ID2EXEREG.JAVA)

输入信号量：

类型	名称	作用
int	RegValIn1	第一个寄存器的值
int	RegValIn2	第二个寄存器的值
int	RegAddrIn1	第一个寄存器的地址
int	RegAddrIn2	第二个寄存器的地址
Boolean	RegEIn1	第一个寄存器是否有效
Boolean	RegEIn2	第二个寄存器是否有效
int	ImIn	立即数的值
int	RegWAddrIn	要写入的寄存器地址
int	ALUCtrlIn	ALU 控制码
Boolean	MemWEIn	内存写使能
Boolean	RegWEIn	寄存器写使能
int	CChoALUIn2	ALU 第二输入选择器控制码

int	CChoRegWValIn	寄存器写回选择器控制码
Boolean	islwswIn	是否为 lwsw 类型指令
Boolean	reset	同步复位

输出信号量

类型	名称	作用
int	RegValOut1	第一个寄存器的值
int	RegValOut2	第二个寄存器的值
int	RegAddrOut1	第一个寄存器的地址
int	RegAddrOut2	第二个寄存器的地址
Boolean	RegEOut1	第一个寄存器是否有效
Boolean	RegEOut2	第二个寄存器是否有效
int	ImOut	立即数的值
int	RegWAddrOut	要写入的寄存器地址
int	ALUCtrlOut	ALU 控制码
Boolean	MemWEOut	内存写使能
Boolean	RegWEOut	寄存器写使能
Boolean	CChoALUOut2	ALU 第二输入选择器控制码
int	CChoRegWValOut	寄存器写回选择器控制码
Boolean	islwswOut	是否为 lwsw 类型指令

主要功能：ID 与 EXE 阶段之间的寄存器，用于暂时保存流水线后续阶段需要的数据和信号。由于是 control 单元之后的寄存器，这个单元保存的信号较多。

CHOALU1 (ORG.KDE9.OTHERS/CHOALU1.JAVA)

输入信号量：

类型	名称	作用

int	RegVal	寄存器值
int	ALUVal1	ALU 运算结果（MEM）
int	ALUVal2	ALU 运算结果（WB）
int	TChoALU1	Transfer 对选择器的控制码

输出信号量

类型	名称	作用
int	A	ALU 第一输入

主要功能：ALU 第一输入数据选择器，其控制码由转发单元给出，以便在寄存器堆给出的第一个寄存器值和两个转发回路之间选择，解决数据冲突。

CHOALU2 (ORG.KDE9.OTHERS/CHOALU2.JAVA)

输入信号量：

类型	名称	作用
int	RegVal	寄存器值
int	ImVal	立即数值
int	ALUVal1	ALU 运算结果（MEM）
int	ALUVal2	ALU 运算结果（WB）
boolean	CChoALU2	Control 对选择器的控制码
int	TChoALU2	Transfer 对选择器的控制码

输出信号量

类型	名称	作用
int	b	ALU 第二输入
int	RegVal2	寄存器值输出

主要功能：ALU 第二输入数据选择器，其控制码由转发单元和 control 单元共同给出，以便在寄存器堆给出的第二个寄存器值、指令立即数值和两个转发回路之间选择，解决数据冲突。

ALU (ORG.KDE9.ALU/ALU.JAVA)

输入信号量：

类型	名称	作用
int	a	ALU 第一输入
int	b	ALU 第二输入
int	Q	ALU 控制码
boolean	Cin	上一级进位，这里恒为 0

输出信号量

类型	名称	作用
int	ret	ALU 运算结果
Boolean	T	T 标志位

主要功能：主要的运算单元，可以处理的运算包括 add、sub、and、or、not、xor、sll、srl、sra、有符号比较和无符号比较等。

EXE2MEMREG (ORG.KDE9.REGISTER/EXE2MEMREG.JAVA)

输入信号量：

类型	名称	作用
int	ALUValIn	ALU 的运算结果
int	RegValIn2	第二寄存器的值
boolean	TIn	T 标志位的值
int	RegWAddrIn	要写入寄存器的地址
boolean	MemWEIn	内存写使能

boolean	RegWEIn	寄存器写使能
int	CChoRegWValIn	寄存器写回数据选择器
boolean	islwswIn	是否为 lwsw 类型指令
boolean	reset	同步复位

输出信号量

类型	名称	作用
int	ALUValOut	ALU 的运算结果
Int	RegValOut2	第二寄存器的值
boolean	TOut	T 标志位的值
int	RegWAddrOut	要写入寄存器的地址
boolean	MemWEOut	内存写使能
boolean	RegWEOut	寄存器写使能
int	CChoRegWValOut	寄存器写回数据选择器
boolean	islwswOut	是否为 lwsw 类型指令

主要功能：EXE 与 MEM 阶段之间的寄存器，用于暂时保存流水线后续阶段需要的数据和信号。

MEM2WBREG (ORG.KDE9.REGISTER/MEM2WBREG.JAVA)

输入信号量：

类型	名称	作用
int	ALUValIn	ALU 的运算结果
int	MemValIn	内存读出的数据
boolean	TIn	T 标志位的值
int	RegWAddrIn	要写回的寄存器地址
boolean	RegWEIn	寄存器写使能
int	CChoRegWValIn	寄存器写回数据选择器控制码

boolean	reset	同步复位

输出信号量

类型	名称	作用
int	ALUValOut	ALU 的运算结果
int	MemValOut	内存读出的数据
boolean	TOut	T 标志位的值
int	RegWAddrOut	要写回的寄存器地址
boolean	RegWEOut	寄存器写使能
int	CChoRegWValOut	寄存器写回数据选择器控制码

主要功能：MEM 与 WB 阶段之间的寄存器，用于暂时保存流水线后续阶段需要的数据和信号。

CHOREGWVAL (ORG.KDE9.OTHERS/CHOREGWVAL.JAVA)

输入信号量：

类型	名称	作用
int	ALUVal	ALU 运算结果
int	MemVal	内存读出的数据
boolean	T	T 标志位的值
int	CChoRegWVal	Control 单元给出的控制码

输出信号量

类型	名称	作用
int	RegWVal	要写回的寄存器值

主要功能：寄存器写回数据选择器，由 control 单元给出控制码选择内存返回值或 ALU 运算结果之一作为写回值。

TRANSFER (ORG.KDE9.OTHERS/TRANSFER.JAVA)

输入信号量：

类型	名称	作用
int	RegWAddr1	寄存器写地址（MEM）
int	RegWAddr2	寄存器写地址（WB）
boolean	RegWE1	寄存器写使能（MEM）
boolean	RegWE2	寄存器写使能（WB）
int	RegAddr1	第一寄存器地址（EXE）
boolean	RegAddrE1	第一寄存器是否有效
int	RegAddr2	第二寄存器地址（EXE）
boolean	RegAddrE2	第二寄存器是否有效

输出信号量

类型	名称	作用
int	TChoALU1	对 ALU 第一输入选择器的控制
int	TChoALU2	对 ALU 第二输入选择器的控制

主要功能：转发单元，负责给出 ALU 两个数据选择器的控制码，并与 control 单元控制码综合控制 ALU 输入，以便解决运算过程中的数据冲突。

TRANSFERCHOPCVAL (ORG.KDE9.TRANSFERCHOPCVAL.JAVA)

输入信号量：

类型	名称	作用
int	RegVal	寄存器值
int	ALUVal1	ALU 运算结果（EXE）
int	ALUVal2	ALU 运算结果（MEM）
int	RegWAddr1	寄存器写地址（ID）
boolean	RegWE1	寄存器写使能（ID）

int	RegWAddr2	寄存器写地址（EXE）
boolean	RegWE2	寄存器写使能（EXE）
int	RegAddr	要读的寄存器地址（EXE）

输出信号量

类型	名称	作用
int	Valx	输出的寄存器值

主要功能：转发单元，负责给 PC 输入数据选择器提供正确的寄存器值，实现 PC 值的快速返回，以便解决跳转语句中出现的数据冲突。

CHOPCCTRL (ORG.KDE9.OTHERS/CHOPCCTRL.JAVA)

输入信号量：

类型	名称	作用
int	Im	立即数值
int	Val	寄存器值
int	PC	当前处于 IF 阶段指令的 PC 值
int	PCx	当前处于 ID 阶段指令的 PC 值
int	CChoPCCtrl	Control 单元给出的控制码

输出信号量

类型	名称	作用
int	lastPC	送给 PCUnit 的输入值

主要功能：PC 输入数据选择器，提供对跳转语句的支持，使程序能够正确运行。

该模块通过 control 单元给出的控制码，选择寄存器值、立即数值、IF 阶段指令的 PC 或 ID 阶段指令的 PC 之一提供给 PCUnit 单元。

存储器部分

存储器部分主要有两个组件：

1. Cache (org.kde9.memory/Cache.java)
2. Memory (org.kde9.memory/Memory.java)

考虑到执行效率和开销的等因素，我们没有采用大数组的方式实现 Cache 和 Memory，而是通过 Hash 表的方式实现，Memory 和 Cache 的大小是根据使用情况动态变化的，这样既做到了快速查找又可以尽可能的节约系统开销。不同的是，Cache 有最大容量限制，而 Memory 没有。

通过在 Cache 内建立引索，实现 Cache 替换的最久未使用原则。通过数据 Cache 的缓存，实现数据写回的 Cache 替换延迟写回原则，这样可以尽可能的减少 Memory 的访问次数，提高流水线的效率。

编译器部分

使用三遍扫描编译的方法，对文本格式的汇编程序进行处理。编译器的主要组成：

1. 外壳 (org.kde9.compile/Compiler.java)
2. 第一第二遍扫描 (org.kde9.compile/Standardization.java)
3. 第三遍扫描 (org.kde9.compile/Check.java)

第一遍和第二遍扫描将对程序中出现的标号进行处理，通过计算指令的条数将所有标号替换为立即数，并将程序代码转换为中间代码形式。中间代码表示是将指令转化为指令编码、寄存器标号、立即数的表示形式。

第三遍扫描将中间代码表示形式转化为二进制代码形式，并对指令中出现的错误进行简单的判断。

可以判断的错误类型有：

1. 不存在的指令。
2. 寄存器编号错误。

3. 指令格式错误。
4. 寄存器格式错误。
5. 立即数格式错误。

CPU 框架图