CompArch Lab1 Report

PB20111686 黄瑞轩

1 描述执行一条 XOR 指令的过程

- IF 阶段
    - 根据 IF PC 的值从 Instruction Cache 中获取 XOR 指令
    - 根据 NPC 的控制信号和内容更新 IF PC
    - 将指令和 PC+4 存入 IF/ID 段间寄存器（IR 和 ID PC）
- ID 阶段
    - 从 IF/ID 段间寄存器（IR）获得指令
    - 对 XOR 指令译码，获得 rs1, rs2 和 rd 的标号、Controller 控制信号以及 Hazard Detection 控制信号
    - 通过 rs1, rs2 读寄存器，将这两个寄存器的内容、rs1, rs2 和 rd 的标号以及 Controller 控制信号存入 ID/EX 段间寄存器
- EX 阶段
    - 从 ID/EX 段间寄存器取得控制信号、寄存器标号和寄存器内容
    - 由于可能出现 Load-Use 数据冒险，ALU 在获取操作数时是根据 Hazard Detection 的 OpSel 信号选择寄存器内容还是 WB 段的输出或 ALU 的输出
    - 根据 ALU Func 控制信号，控制 ALU 使用寄存器内容进行异或运算，XOR 指令的两个操作数都是寄存器，不发生冒险时 ALU 的操作数来源均为寄存器内容
    - 将 ALU 的结果保存在 EX/MEM 段间寄存器（RESULT），传递 MEM 阶段仍需要的 Controller 控制信号和 rd 标号
- MEM 阶段
    - 从 EX/MEM 段间寄存器取得控制信号、rd 标号和 ALU 的输出（RESULT）
    - 将 RESULT 内容保存在 MEM/WB 段间寄存器（WBDATA），传递 WB 阶段仍需要的 Controller 控制信号和 rd 标号
- WB 阶段
    - 从 MEM/WB 段间寄存器取得控制信号、rd 标号和 WBDATA
    - 根据控制信号写寄存器，指令完成

2 描述执行一条 BEQ 指令的过程

- IF 阶段
    - 根据 IF PC 的值从 Instruction Cache 中获取 BEQ 指令
    - 根据 NPC 的控制信号和内容更新 IF PC
    - 将指令和 PC+4 存入 IF/ID 段间寄存器（IR 和 ID PC）
- ID 阶段
    - 从 IF/ID 段间寄存器（IR）获得指令和 ID PC 值
    - 对 BEQ 指令译码，获得 rs1, rs2 和偏移量、Controller 控制信号以及 Hazard Detection 控制信号
    - 通过 rs1, rs2 读寄存器，获得寄存器内容
    - 通过 Immediate Generate 获得符号扩展过的偏移量，左移一位后通过一个单独的加法器和 ID PC 相加获得分支目标地址
    - 将寄存器的内容、rs1, rs2 的标号、分支目标地址以及 Controller 控制信号存入 ID/EX 段间寄存器
- EX 阶段
    - 从 ID/EX 段间寄存器取得控制信号、寄存器标号、分支目标地址和寄存器内容
    - 由于可能出现 Load-Use 数据冒险，ALU 在获取操作数时是根据 Hazard Detection 的 OpSel 信号选择寄存器内容还是 WB 段的输出或 ALU 的输出
    - 根据 BrType 控制信号，Branch Module 使用寄存器内容进行相等判断运算，BEQ 指令的两个操作数都是寄存器，不发生冒险时 ALU 的操作数来源均为寄存器内容
    - 根据 ALU 的计算结果，控制 IF 阶段 NPC 选择器将此时的分支目标地址作为 NPC，指令完成

3 描述执行一条 LHU 指令的过程

- IF 阶段
    - 根据 IF PC 的值从 Instruction Cache 中获取 LHU 指令
    - 根据 NPC 的控制信号和内容更新 IF PC
    - 将指令和 PC+4 存入 IF/ID 段间寄存器（IR 和 ID PC）
- ID 阶段
    - 从 IF/ID 段间寄存器（IR）获得指令
    - 对 LHU 指令译码，获得 rs1 和 rd 的标号、Controller 控制信号以及 Hazard Detection 控制信号
    - 通过 rs1 读寄存器，获得寄存器内容
    - 通过 Immediate Generate 获得符号扩展过的偏移量

    - 将这 rs1 内容、rs1 和 rd 的标号、偏移量以及 Controller 控制信号存入 ID/EX 段间寄存器
- EX 阶段
    - 从 ID/EX 段间寄存器取得控制信号、寄存器标号、偏移量和寄存器内容
    - 根据 Controller 控制信号，选择 ALU 的两个操作数分别为 rs1 内容和偏移量
    - 根据 ALU Func 控制信号，控制 ALU 使用寄存器内容进行相加运算
    - 将 ALU 的结果保存在 EX/MEM 段间寄存器（RESULT），传递 MEM 阶段仍需要的 Controller 控制信号和 rd 标号
- MEM 阶段
    - 从 EX/MEM 段间寄存器取得控制信号、rd 标号和访存地址
    - 根据访存地址和 Load TypeM 控制信号获得无符号扩展的半字，WB Select 控制选择访存内容写入 WBDATA
    - 将 rd 标号和 WB 阶段还需要的 RegWrite 信号存入 MEM/WB 段间寄存器
- WB 阶段
    - 从 MEM/WB 段间寄存器取得控制信号、rd 标号和 WBDATA
    - 根据控制信号写寄存器，指令完成

4 如果要实现 CSR 指令（csrrw，csrrs，csrrc，csrrwi，csrrsi，csrrci），设计图中还需要增加什么部件和数据通路？给出详细说明。

各 CSR 指令的行为如下：

- csrrw 指令：t = CSRs[csr]; CSRs[csr] = x[rs1]; x[rd] = t
- csrrwi 指令：t = CSRs[csr]; CSRs[csr] = zimm; x[rd] = t
- csrrs 指令：t = CSRs[csr]; CSRs[csr] = x[rs1] | t; x[rd] = t
- csrrsi 指令：t = CSRs[csr]; CSRs[csr] = zimm | t; x[rd] = t
- csrrc 指令：t = CSRs[csr]; CSRs[csr] = ~x[rs1] & t; x[rd] = t
- csrrci 指令：t = CSRs[csr]; CSRs[csr] = ~zimm & t; x[rd] = t

可以抽象为三件事：

- 保存旧的 CSR 寄存器值
- 用立即数/x[rs1] 与旧 CSR 值做某种运算，并将结果写入 CSR 寄存器
- 写回，令 x[rd] 为旧 CSR 值

要增加的部件

- 因为 CSR 指令涉及对 CSR 寄存器的读写，所以我们需要在设计图中增加 CSR 寄存器部件

- 对段间寄存器需要做如下修改
    - 因为要在 ID 阶段读出旧的 CSR 值，并且在 WB 阶段写回，所以 ID/EX、EX/MEM 段间寄存器均需增加 64 位 OldCSR 域和 12 位 csr 域进行值的直接传递
    - 在 EX 阶段计算之后，需要在 MEM 阶段将计算结果写入 CSR 寄存器，因此 EX/MEM 段间寄存器增加 64 位 ResultCSR 域
    - 在译码阶段要知道这条指令是否是 CSR 指令，增加 1 位 IsCSR 域
    - 增加译码 CSR 指令时的 ALUFunc 逻辑
    - 扩展 Op1、Op2 和 WBSelect 信号，使之支持 4 路多选

要增加的数据通路

- IF 段不做特别的修改
- 在 ID 段，用译码得到的 csr 读取 CSR 寄存器，将结果存在 ID/EX.OldCSR 中
- 在 EX 段，根据 Op1、Op2 控制信号选择操作数。如非 i 指令时 Op1 选择 x[rs1]，Op2 选择 ID/EX.OldCSR；i 指令时 Op1 选择 ID/EX.OldCSR，Op2 选择 IMM。根据 ALUFunc 进行运算，并将运算结果存在 EX/MEM.ResultCSR 中
- 在 MEM 段，根据 EX/MEM.csr 将 EX/MEM.ResultCSR 中的内容写到 CSR 寄存器中，并由 WBSelect 控制将 EX/MEM.OldCSR 写入 MEM/WB.WBDATA 中
- WB 段不做特别的修改

5 Verilog 如何实现立即数的扩展?

零扩展：使用 `{}` 符号，如 imm[5:0] 零扩展到 32 位，则为 `{ 26'b0, imm }`

符号扩展：使用 `imm[5]` 获得 imm 的符号位，`{{ 26{ imm[5] }}, imm }`

6 如何实现 Data Memory 的非字对齐的 Load 和 Store?

如果是在一个字以内的访存：

Load：通过 LoadTypeM 控制信号，将从 Data Memory 中读出的内容经过 Data Extension 模块之后即可实现非字对齐的 Load

Store：从设计图来看，如果 CacheWrite 信号是 1 位的，似乎无法做到。如果 CacheWrite 像 LoadTypeM 那样能够指示写入的方式，并且 WData 还需要根据 CacheWrite 做相应的 Data Extension，那么就可以类似地做到非字对齐的 Store

如果是多个字的访存，则需要多个时钟周期来完成访存。

7 ALU 模块中，默认 wire 变量是有符号数还是无符号数?

在 Verilog 中，默认 wire 变量是无符号数，如果需要使用有符号数，需要使用 `signed` 关键字。

8 简述 BR 信号的作用。

BR 信号作为 NPC 多选器的选择信号，可以将 NPC 选定为 BR 指令所指向的分支目标地址。

9 NPC Generator 中对于不同跳转 target 的选择有没有优先级?

因为 JAL 指令是在 ID 段就判断，而 JALR、BR 都是在 EX 段才判断，所以这两套指令之间有优先级。如果后者有效而前者也有效，说明后者在前者之前，此时应当执行 BR 或者 JALR 指令。

10 Harzard 模块中，有哪几类冲突需要插入气泡，分别使流水线停顿几个周期?

Hazard 模块可以检测**数据冒险 Load-Use 冲突**和**控制冒险**两类冲突。

- 对于前者，流水线需要暂停一个周期，在 Load MEM 阶段后就可将数据给 stall 一个周期后的 Use EX 阶段
- 对于后者，流水线需要暂停两个周期，因为在分支指令 EX 阶段计算分支是否发生以及分支地址之后，才可以进入下一条指令的 IF 阶段取指令

11 Harzard 模块中采用静态分支预测器，即默认不跳转，遇到 branch 指令时，如何控制 flush 和 stall 信号?

如果使用这种做法，不需要控制 stall 信号，也即不需要暂停流水线。

当 branch 指令到达 MEM 阶段时，如果预测失败，此时将 FlushD, FlushE 置位，清除 ID 和 EX 阶段的指令。

12 0 号寄存器值始终为 0，是否会对 forward 的处理产生影响?

会，比如使用 `addi x0, x1, 2` 这样的指令，需要避免 forward 非零的结果。

修改方案：在判断冒险条件时，额外判断 EX/MEM.rd $\neq 0$ 或 MEM/WB.rd $\neq 0$。