RISC-V CPU report

冯思远
516030910575

1 概要

这是 2017 计算机系统 (I) 大作业的报告文档。项目使用 Verilog 语言，基于 FPGA 芯片，实现一个简化的 RISC-V 指令集 CPU。并基于 uart 通信协议，与 PC 端的内存系统实现交互。但由于时间较紧，项目未最终完成。如无特殊说明，下文为针对目前进度的报告。

2 计算核心实现

项目采用标准五级流水架构，即取指、译码、执行、访存、回写五级流水线。为缓解因 RAW 产生的 stall 节拍，增加 forwarding 操作。分别从执行和访存阶段 forwarding 到译码阶段。经过优化过后仅当后一条指令对前一条存在数据依赖，且前一条指令为 load 指令，才会 stall 一拍，相比原始五级流水有了很大的提升。

3 内存实现

原计划需要将内存实现在 pc 端，并由 uart 通信。由于时间原因未达到该程度，现阶段采用模拟内存的方式，并采用 uart 方式进行通信，尽可能仿真真实情况，并为下一阶段提供经验。内存控制模块采用张哲恺助教的代码，详见https://github.com/sxtyzhangzk/mips-cpu/

4 cache 实现

由于 uart 通信速度实在太慢，急需 cache 进行加速。项目采用 2 路组关联 cache，单路采用块大小为 16Byte 的 64 行直接映射 cache。采用块状读写，即每次读取一整块，即 16 个 Byte。

4.1 通信速度

根据通信协议，每次读取需要 4∗2+5+19 = 32 个 uart 周期，而单个读取需要 (4∗2+5+5)∗4 = 72，提速 100% 以上。即使有 50% 的数据没有被利用，也比单个读取要快。

4.2 块大小

经过研究，将块大小定位 16Byte，原因如下：（下文提到的周期均为传输一个 8bit 数据的 uart 周期）

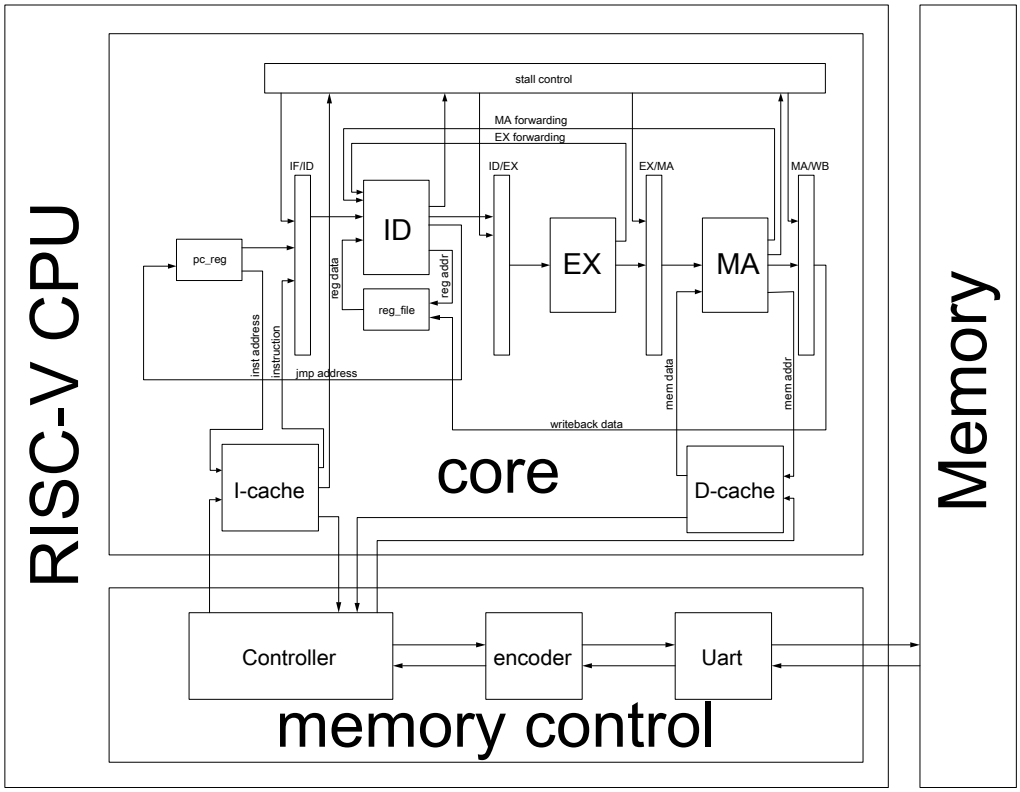
1. 由通信速度看，块大小越大，单个数据的平均用时就少。但是这个变化随着块大小的增大而减小。当块大小为 32 时，平均仍需要 1.5 个周期，未比块大小 16 的 2 周期增加很多。

2. 按块读取虽然快，但是如果仅需要少数数据，则会造成大量无效数据的读写。程序有本地性，但是没有本地性的内存，很多情况是仅有一个数据有效，那么造成的代价是 $50 - 18 = 32$ 个周期，远大于块大小为 16 时的 $32 - 18 = 14$ 个周期。

3. 在 cache 大小固定的情况下，块大小越大，代表着行数越小，越容易被替换。而替换的代价是很大的，如果被修改过需要 2 倍于读入的周期。尽可能减少替换可以减少时间。

5 附录

5.1 架构图



5.2 参考资料

1. 助教代码：https://github.com/sxtyzhangzk/mips-cpu/

2. 自己动手写 CPU（雷思磊，电子工业出版社）

5.3 特别感谢

提供测试代码的范舟同学和给予我很多帮助的吴章昊同学

5.4 Github 地址

https://github.com/Hzfengsy/RISC-V