COMPUTER ORGANIZATION PROJECT REPORT

计算机组成原理大实验报告

任一　黄嘉良　潘传宇
清华大学　计算机科学与技术系

December 11, 2020

1 实验目标概述

本实验中我们的目标是实现一个支持中断异常和页表 TLB 的多周期处理器。特别的，我们的处理器可以运行含中断异常和页表的监控程序，能够针对错误指令、页表异常等进行异常处理，能够利用 TLB 加速基于页表的内存读写。

2 主要模块设计

2.1 Data Path

我们设计的数据通路如下图所示。数据通路中包含的主要模块有 PC/PCnow 寄存器、MMU、IR/DR 寄存器、立即数生成器 immGen、普通寄存器堆 RegFile、AR/BR/CR 寄存器、ALU、控制寄存器、控制模块 controller 等。数据通路图中，“红线”为 controller 接出的控制信号，“绿线”为接入 controller 的反馈信号，“蓝线”为模块间的数据传输信号。

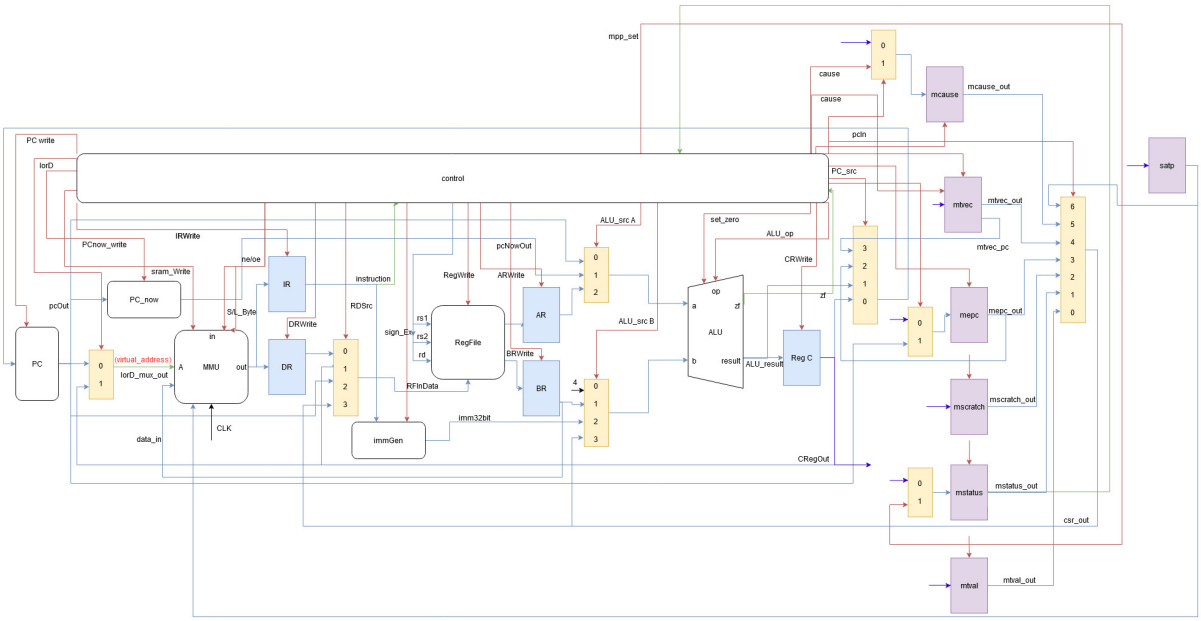

Figure 1: 数据通路

以 add 指令为例，首先由 PC 模块确定当前的指令地址，在取指阶段将该地址送入 MMU 模块、将指令读出并存入 IR 寄存器中。进入译码阶段，指令通过数据通路进入 controller 和 immGen，由 controller 解析指令并生成源、目的寄存器并送入 RegFile，由 immGen 生成立即数送入 b 选择器。从寄存器堆中取出对应寄存器中存储的值后将存入 AR、BR 寄存器，并由 a 选择器和 b 选择器来确定传入 ALU 的值，具体到 add 指令来说，

a 选择器选择 AR 的值，b 选择器选择 BR 的值。在执行阶段 ALU 对 a、b 输入执行加法运算，并输出结果（ALUresult）到 RegC 和 pc 选择器中。由于 add 指令没有访存阶段，因此不需要将地址或数据送回 MMU。在最后的写回阶段，ALU 的运算结果通过数据通路进入 RegFile 并存入。

除了上述比较基本的模块和通路以外，为了支持终端异常和页表，我们还在数据通路中加入了各种控制寄存器。比如，在发生中断异常进入机器模式时，当前 PC 值会通过数据通路存入 mepc 寄存器，并将 mtvec 寄存器的值赋给 PC，使其跳转至异常处理程序；以上对于 PC 的操作通过 PC 选择器实现。

2.2 Controller

在我们的多周期处理器中，Controller 会根据处理器处于的周期，给 CPU 其他模块相应的控制信号。

以 lw 这条指令为例。lw 指令需要经过 IF, DEC, EXE, MEM, WB 这些周期。在 IF 阶段，lw 指令需要从内存中读取出指令，放到 IR 寄存器中。因此 Controller 需要让内存模块进行读内存的操作、打开 IR 寄存器的写使能，这样就能把指令读到内存当中。又如在 EXE 周期，lw 指令需要用基址寄存器的值加上符号扩展的立即数的值，得到待取指的内存位置。因此在 EXE 周期，Controller 需要把 ALU 的操作设置为加法，并且让符号扩展单元的符号扩展方式设置为 lw 指令格式所对应的符号扩展方式。

2.3 MMU

在支持了页表之后，一方面为了加速页表的查询，另一方面为了统一上层的 CPU 和内存的交互接口，引入了内存管理单元 (MMU)。

关于内存管理单元的主要设计如下：

1. 实现了表项大小为 1 的 TLB。
2. TLB 的表项中有一个 1 位的 Valid 字段，当执行 sfence.vma 这条指令时，会将 valid 字段置 0，表示该条 TLB 表项无效。在判断 TLB 是否匹配时，也会加入对 Valid 字段检查。
3. 所有对于内存的访问（包括串口），都会经过内存管理单元，统一 CPU 读写数据的接口。
4. 实现了 PageFault 异常。查询一级页表时，会检查 V 字段；当读内存时，会检查二级页表中的 V 和 R 字段；当写内存时，会检查二级页表中的 V, R 以及 W 字段。

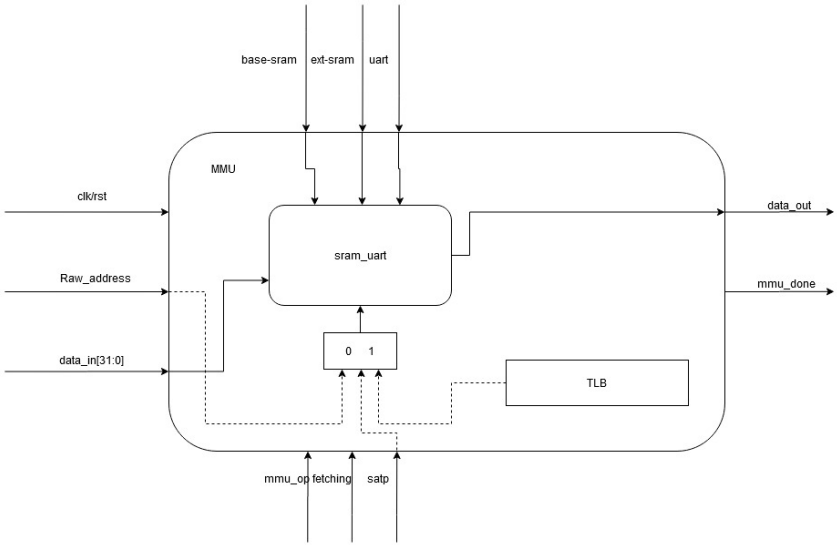

Figure 2: 内存管理单元

5. MMU 的状态机方面除了共有的 Done，Idle 以及读写等状态外，读第一级页表、读第二级页表、更新 TLB 表项以及执行 sfence 分别独立为一个周期

3 实验成果展示

我们小组录制了展示的视频。
视频链接是`https://www.bilibili.com/video/BV1hi4y157Hh/`
源码链接是`https://github.com/nmrenyi/RISCV-MultiCycle-CPU`

4 实验心得体会

4.1 黄嘉良

计原大实验，锻炼了我多方面的能力。我负责的内容主要是内存管理单元，涉及到比较麻烦的页表读取、虚实地址转换。首先是对 RICSV 标准的阅读，了解 RICSV 对于页表的组织、虚拟地址的标准等。另一方面是标准到最后实现的转换，我们并不必、也无法实现出标准的 RISCV 要求的版本，我们需要结合监控程序和其余模块（包括异常的实现）来调整我们的设计。然后是组内的交流合作能力，队友之间互相明确每一阶段的实现目标。最后是对于硬件和硬件描述语言的理解，在之前的数字逻辑课程中对于这方面的理解并不清晰，经过大实验后，明确了许多硬件设计的原理和理念，对于计算机结构有了更深的认识。

4.2 潘传宇

本实验中我主要负责数据通路的设计和调试，以及中断异常的相关实现。通过本实验我对计算机的硬件实现有了更深刻的认识，加深了对于多周期 cpu 的理解。通过本实验我掌握了更加标准、更加系统的硬件设计方式，对于 Verilog 语言的掌握更加熟练。通过阅读 RISCV 官方文档，尤其是中断异常部分，使我对于计算机中断异常处理机制有了更加清楚的认识。值得一提的是，在我们代码编写的过程中遇到了大大小小的问题，有的可以通过检查代码解决，但有的只能通过仿真调试，而在仿真调试的过程中，我对于硬件时序的理解更进了一步。总之，这是一次非常宝贵的硬件实践经历，在这一过程中我收获颇丰。

4.3 任一

本次实验中，我主要负责多周期 CPU 跑通基础版本监控程序的实现。作为较早开始做基础版本实现的同学，我在整个实现特别是调试的过程中遇到了很多的问题。在这个过程中，我在课程交流群中提出了很多的问题，也获得了来自老师、助教和同学们的非常多帮助，经过锲而不舍的 coding, debugging, communicating, 我终于让小组的 CPU 顺利跑通了基础版本的监控程序。在跑通之后，我也积极维护答疑文档，把自己调试过程中遇到的有借鉴价值的问题，整理到文档中，希望帮助到更多的同学。

在后期中断、异常和 TLB 的实现中，我的队友做了主要的实现，我负责协助调试、重构基础版本代码等工作。整个团队合作非常愉快，我也更加深入地理解了计算机 CPU 的设计，对提高系统思维能力很有帮助。