组队大实验报告

计04 岑奇航 计科01 玄镇 计05 刘好

实验目标

1. 深入理解流水线结构计算机指令的执行方式，掌握流水处理器的基本设计方法。
2. 深入理解计算机的各部件组成及内部工作原理。
3. 加深对于 RV32I 指令集的理解。
4. 掌握计算机外部输入输出的设计。
5. 提高硬件设计和调试的能力。

实验内容

1. 能够支持监控程序基本版本用到的 RV32I 指令集，以及每小组的额外三条指令。
2. 具有SRAM访问功能，能够满足监控程序数据与代码的存储需求。
3. 利用串口实现计算机的输入输出模块，能够支持监控程序与 PC 的相互通信。
4. 实现中断处理机制，对串口产生的中断信号，运行中断处理程序接收数据。
5. 支持虚拟内存管理，分离用户程序和监控程序内核的地址空间。
6. 实现时钟中断、S 态，完善异常处理，支持完整的 rv32i 指令，最终支持uCore的运行
7. 实现TLB，减少启用页表下访存的周期数。

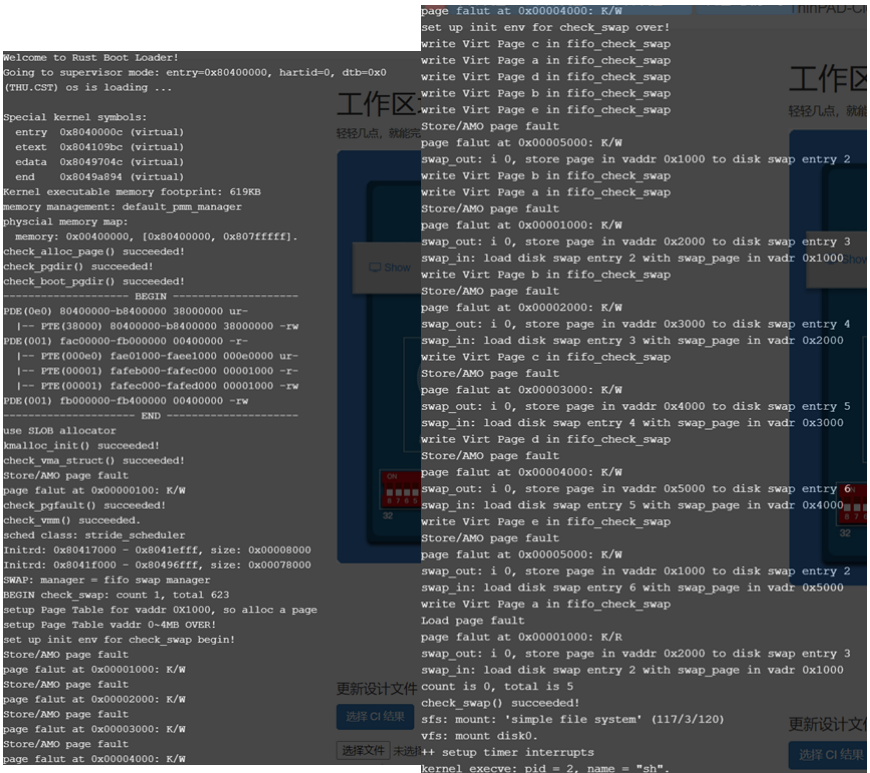
效果展示

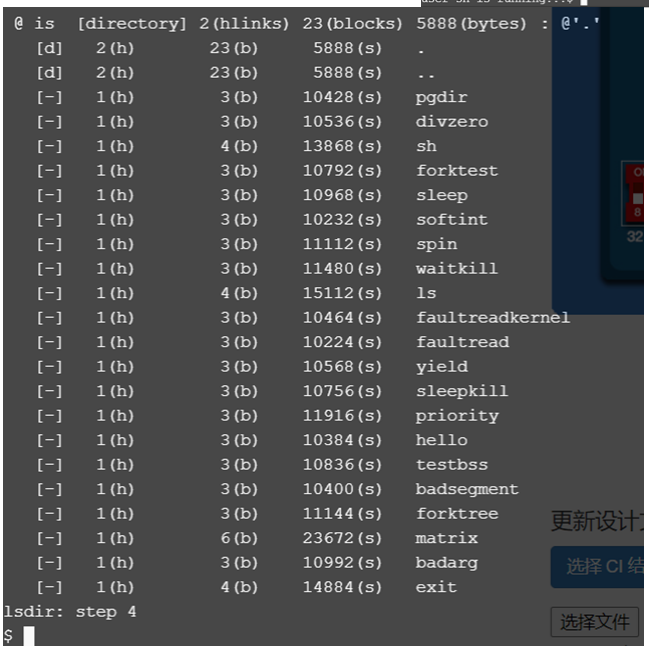
监控程序v2-v3

```
>> G
addr: 0x800010c0
killed timeout program.

elapsed time: 0.381s
>> G
addr: 0x800010a8
OK
elapsed time: 0.000s
>>
```

```
>> F
>>file name: D:\Acquah\ucore\supervisor-rv\test.s
>>addr: 0x80100000
reading from file D:\Acquah\ucore\supervisor-rv\test.s
[0x80100000] .section .text
[0x80100000] .globl _start
[0x80100000] _start:
[0x80100000] addi t0, x0, 0x1
[0x80100004] addi t1, x0, 0x0
[0x80100008] addi t3, x0, 0xa
[0x8010000c]
[0x8010000c] loop:
[0x8010000c] add t1, t1, t0
[0x80100010] add t0, t0, 0x1
[0x80100014] add t3, t3, -1
[0x80100018] bne t3, x0, loop
[0x8010001c] nop
[0x80100020] jr ra
[0x80100024] nop
>> G
addr: 0x0

elapsed time: 0.000s
>> R
R1 (ra)    = 0x80000430
R2 (sp)    = 0x80000000
R3 (gp)    = 0x00000000
R4 (tp)    = 0x00000000
R5 (t0)    = 0x0000000b
R6 (t1)    = 0x00000037
R7 (t2)    = 0x00000000
R8 (s0/fp) = 0x0000001e
R9 (s1)    = 0x00000000
R10(a0)    = 0x0000004b
R11(a1)    = 0x00000000
R12(a2)    = 0x00000000
R13(a3)    = 0x00000000
R14(a4)    = 0x00000000
R15(a5)    = 0x00000000
R16(a6)    = 0x807fff30
```

uCore





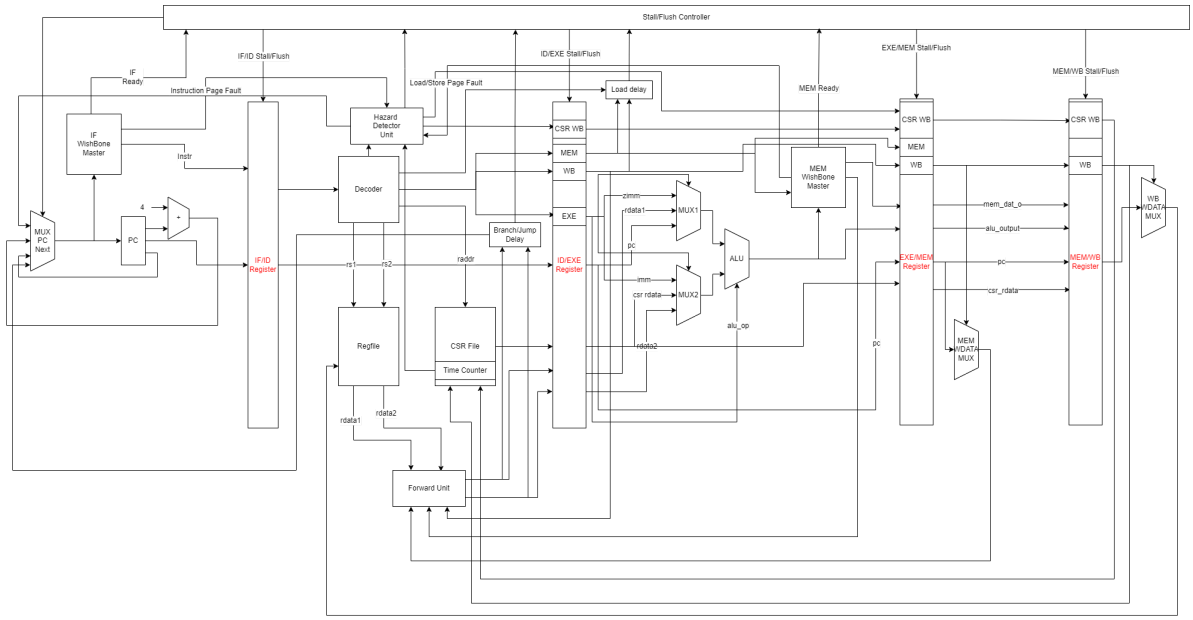
演示视频见`grp19-ucore.mp4`

CPU设计及数据通路图

整个流水线根据一般设计分为5段，即IF、ID、EXE、MEM、WB

- 整个流水线的运行控制受Stall/Flush信号控制，由于访存周期大于1，在IF段以及MEM段访存没结束前设置所有段Stall
- IF段进行指令的取指，由于访问SRAM经过总线，故设计IF段的Wishbone Master与总线交互；为了让IF段开始时便在进行访存，减少周期数，增加了pc_next寄存器用于记录下一条指令地址，将其作为访存地址发入Wishbone Master；结束访存后得到指令，等待流水线暂停结束流入ID段。当异常处理进行时，IF段不取指，更新pc_next并等待异常指令WB段结束后访存。

- ID段进行指令解码，首先判断指令是否引发异常，若是则启动异常处理，设置csr寄存器值，停止流水线直至该指令执行完毕。该阶段同时从寄存器堆中获取寄存器值，并借由Fowarding单元提前更新指令对应寄存器值；由于Forwarding提前至ID段，故可进行Branch/Jump指令的判断，若进行跳转则设置pc_next为跳转地址并flush IF/ID寄存器；若读取的寄存器在EXE段指令中存在写入操作，则引发Load Delay，Stall IF/ID寄存器并Flush ID/EXE寄存器直至写入结束。该段也提前准备好了访存信号以及写回信号，以便后续使用。
- EXE段进行指令执行，主要为将从ID段获取的数据以及解析得到的ALU指令编号输入至ALU中并得到结果。根据指令特点，由ID段对应信号来判断ALU输入选取哪个数据。为了MEM段开始时即进行访存，节省周期，在EXE段将访存信号输入至MEM Wishbone Master中。
- MEM段进行访存，若出现Page Fault则通过异常处理单元清空之前的流水线并停止，抛弃本条指令并设置为epc，设置其他CSR寄存器值并等待执行至WB段结束后重启流水线。若无则等待总线访存完毕获取输出/完成写入，将对应的内容等待交给WB段。
- WB段进行写回，根据信号获取合适的数据，写入寄存器堆中。若之前有异常发生，则修改对应CSR寄存器并发送信号表明异常处理完成，重启流水线。



遇到的困难与解决方案

- 精确异常处理：监控程序v2中的异常只包含ecall, ebreak, mret等指令，均可在ID段进行判断并中止流水线；而实现uCore的过程中需引入Instruction/Load/Store Page Fault以及Time Interrupt，分别由IF段、MEM段以及另外的时钟计数模块触发，epc寄存器的写入值情况更为复杂。在尝试了若干方案并调试后我选择了将IF段的Page Fault以及时钟中断的地址设置为当前ID段PC的下一条指令，即当前IF段PC或者ID段跳转的PC，执行完ID段指令后才重新启动流水线；MEM段Page Fault则直接在MEM段处理，丢弃当前指令并写入epc，执行完当前WB段指令后重启流水线，中间指令全部丢弃。
- CSR寄存器读写问题：在异常发生后如果最后一条指令恰好正在修改csr寄存器，会出现将原本设定好的寄存器的值覆盖，故设定由异常检测单元发出的写入CSR寄存器的信号更加优先
- uCore调试：由于运行uCore时所需时间较长，vivado仿真的速度过慢，基本无法方便的仿真至uCore初始化部分，故配置了ILA并借助线上实验平台进行调试；由于连接问题经常会出现调试时thinpad意外断开连接的情况，故每次触发后将信号保存至本地在进行调试。

心得体会

本次大实验加深了我们对于流水处理器架构的理解，巩固了异常中断处理、虚拟地址和页表、特权态、TLB等知识，并从理论知识出发进行实践。在设计过程中遇到了很多Bug和困难，我们一起合作进行错误的调试与修正，并反过来审视自己出现错误的原因，这一过程增进了我们硬件设计与调试的能力。在进一步实现uCore运行的过程中，我们提高了自己仔细阅读文档的能力，并在文档的指导下完善自己CPU的功能。其间有许多由于理解的错误而产生的Bug，通过重新阅读文档、小组讨论、上板观察现象，我们在合作中克服了这些困难。总的来说，这次大实验让我们受益匪浅。