计算机组成原理 · 大实验报告

第 01 组

计01　于沛楠　2020010877

计01　李宇轩　2020010874

计01　容逸朗　2020010869

实验目标

- 运行 uCore

实验内容

内容简述

基础部分

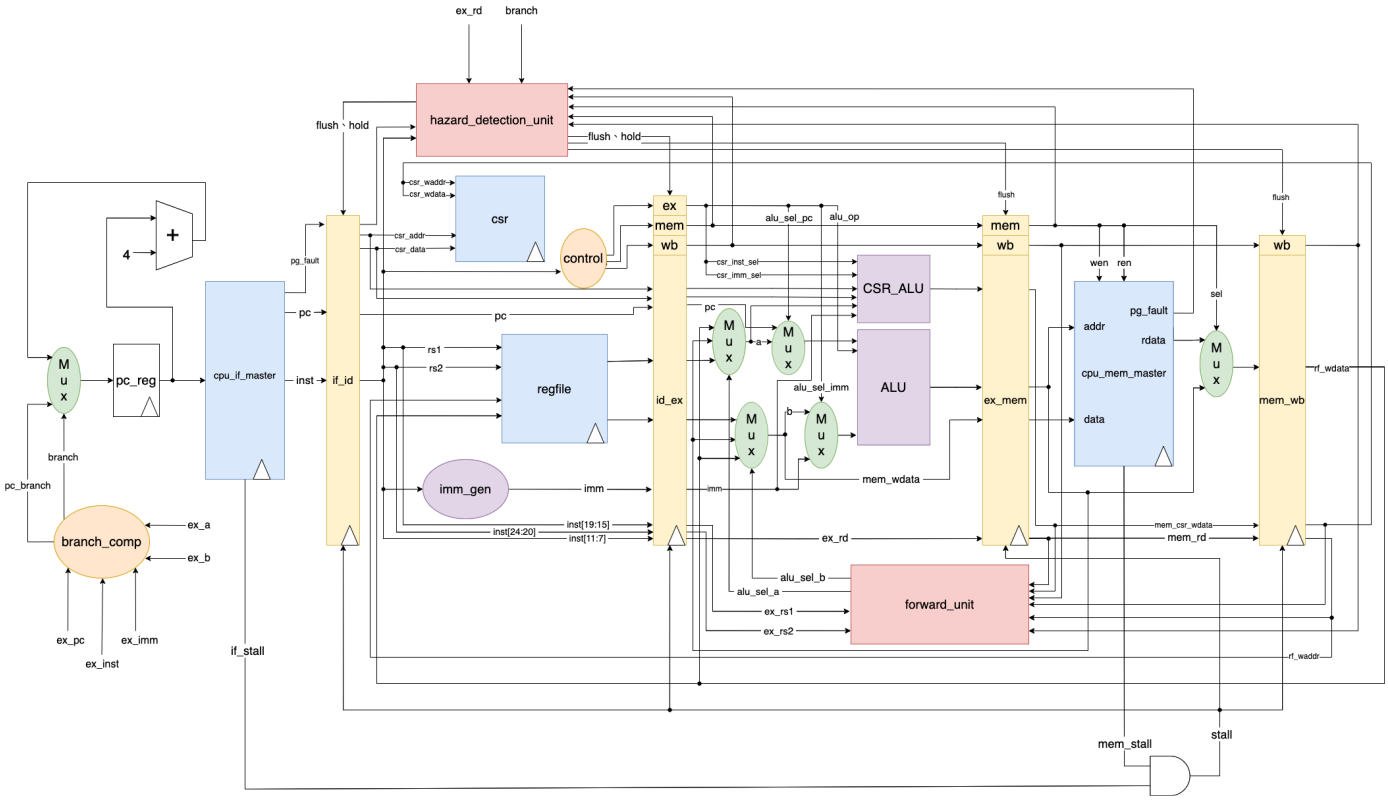
- 能够支持监控程序基本版本用到的 RV32I 指令集。
- 每一个小组还需要实现指定给本小组的额外三条指令。
- 具有内存（SRAM）访问功能，能够满足监控程序数据与代码的存储需求。
- 利用串口实现计算机的输入输出模块，能够支持监控程序与 PC 的相互通信。

提高部分

- 特权态功能补充
- 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。
  - 实现时钟中断
  - 实现 S 态时钟中断
- 异常处理
  - 额外实现三种 Page Fault 异常和 S 态 ECALL 异常
  - 实现中断异常委托：`mideleg`, `medeleg`
- TLB 和虚拟地址
  - 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
- 支持完整的 rv32i 指令
  - RV32I 指令集中除 `FENCE` , `FENCE.TSO` 和 `PAUSE` 外的全部指令
  - Zicsr 扩展中的全部 6 条 CSR 指令
  - Zicntr 扩展（rv-spec 10.1 Base Counters and Timers 章节）中的 `RDTIME` 和 `RDTIMEH` 指令
  - 实现 S 态返回指令 `SRET` 和 `SFENCE.VMA`

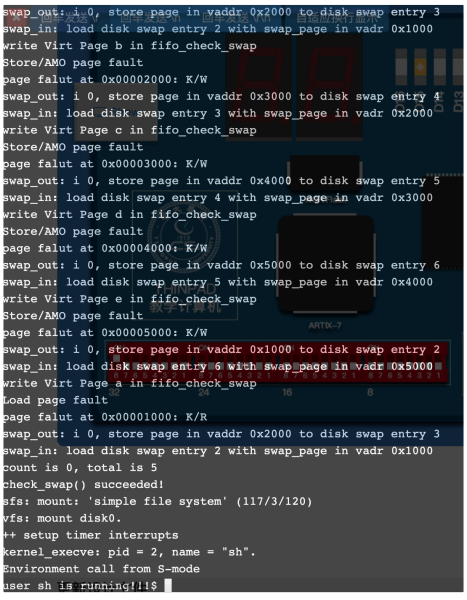
数据通路图

具体设计如下所示：

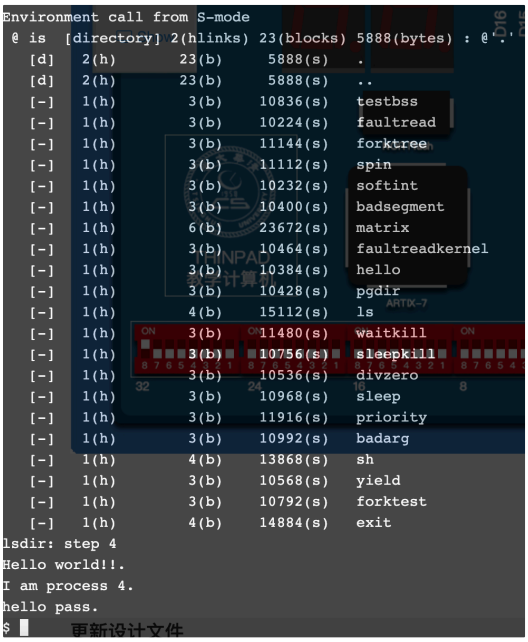


上板截图

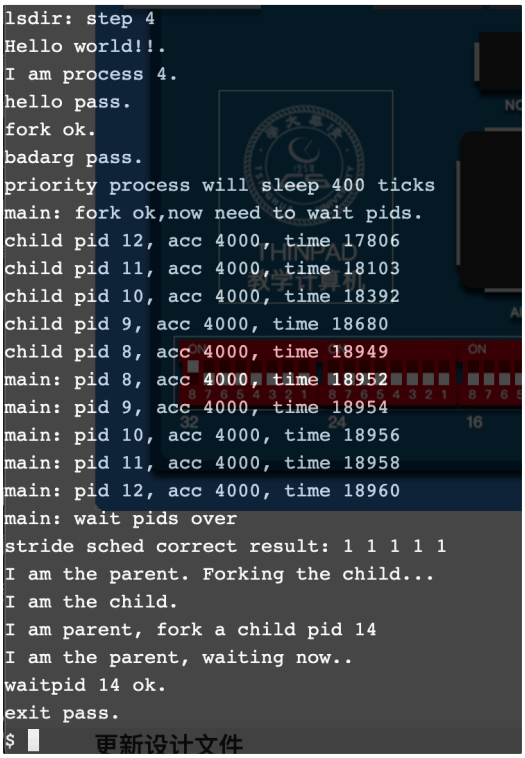
启动



测试 ls


Environment call from S-mode
@ is [directory] 2(hlinks) 23(blocks) 5888(bytes) : @'.'
[d] 2(h) 23(b) 5888(s) .
[d] 2(h) 23(b) 5888(s) ..
[-] 1(h) 3(b) 10836(s) testbss
[-] 1(h) 3(b) 10224(s) faultread
[-] 1(h) 3(b) 11144(s) forktree
[-] 1(h) 3(b) 11112(s) spin
[-] 1(h) 3(b) 10232(s) softint
[-] 1(h) 3(b) 10400(s) badsegment
[-] 1(h) 6(b) 23672(s) matrix
[-] 1(h) 3(b) 10464(s) faultreadkernel
[-] 1(h) 3(b) 10384(s) hello
[-] 1(h) 3(b) 10428(s) pgdir
[-] 1(h) 4(b) 15112(s) ls
[-] 1(h) 3(b) 11480(s) waitkill
[-] 1(h) 3(b) 10756(s) sleepkill
[-] 1(h) 3(b) 10536(s) divzero
[-] 1(h) 3(b) 10968(s) sleep
[-] 1(h) 3(b) 11916(s) priority
[-] 1(h) 3(b) 10992(s) badarg
[-] 1(h) 4(b) 13868(s) sh
[-] 1(h) 3(b) 10568(s) yield
[-] 1(h) 3(b) 10792(s) forktest
[-] 1(h) 4(b) 14884(s) exit
lsdir: step 4
Hello world!!.
I am process 4.
hello pass.
$

部分测例测试


lsdir: step 4
Hello world!!.
I am process 4.
hello pass.
fork ok.
badarg pass.
priority process will sleep 400 ticks
main: fork ok,now need to wait pids.
child pid 12, acc 4000, time 17806
child pid 11, acc 4000, time 18103
child pid 10, acc 4000, time 18392
child pid 9, acc 4000, time 18680
child pid 8, acc 4000, time 18949
main: pid 8, acc 4000, time 18952
main: pid 9, acc 4000, time 18954
main: pid 10, acc 4000, time 18956
main: pid 11, acc 4000, time 18958
main: pid 12, acc 4000, time 18960
main: wait pids over
stride sched correct result: 1 1 1 1 1
I am the parent. Forking the child...
I am the child.
I am parent, fork a child pid 14
I am the parent, waiting now..
waitpid 14 ok.
exit pass.
$

实验总结

困难解决方案

本次实验中，代码查错是我们队伍遇到过的最大问题：

- 首先，每次启动 uCore 都需要十几秒的时间，而 Vivado 的仿真效率不足以支持我们完整观看过程中的所有波形，因此不能利用 Vivado 的仿真功能来调试；
- 在不能仿真的情况下，我们尝试使用静态代码查错的方法，但因为代码体量较大，我们很快打消了这个念头；
- 最终，我们采用了 ILA 在线调试设计的方法，抓取在实际运行时部分关键信号的波形，如每阶段的 pc 信号、重要的 CSR 以及时钟中断部分的内容；
- 通过不断调整测试，最终完成 uCore。

个人心得

于沛楠

通过本次大实验，我深刻地理解了五级流水线处理器的设计方法、流水线中存在的冲突与解决方案、时钟中断、以及 M 态、S 态异常处理的流程。通过阅读 risc-v 文档，我们逐步支持了中断和异常、页表和虚拟地址、以及最终实现了支持 uCore 的处理器，并学会了通过仿真、ILA 等工具对代码进行调试。最后，感谢队友们在实验过程中的付出，感谢助教提供了详尽的实验文档，让我顺利完成了本次大实验。

李宇轩

本次大实验让我收获良多。首先，我基本建立起了硬件编程的思维，打破了软件编程的思维定式。其次我从比较微观的层面理解了计算机组成结构的基本原理，直观体会到数据是怎样在 cpu 上流动的，感受到了流水线在计算机体系结构中的作用。我也深刻意识到在硬件编程的过程中，仔细阅读文档协议，理解硬件原理，耐心仔细仿真的重要性。最后感谢队友的通力协作，让我较为成功地完成了本次大实验。

容逸朗

本次 uCore 实验中，我对操作系统和硬件的配合方式有了更深刻的理解。在最后调试的部分使我明白到仿真对于硬件设计的重要性，这不仅仅是因为以往的软件设计思想套接在硬件上会出现截然不同的反应，还有一方面是因为仿真波形是对于数据流的进一步验证，通过波形，我们可以很直观的找到问题所在。除此之外，实现 uCore 的过程中也让我查阅了不少资料，这使我对 CPU 设计有了更进一步的认识。最后，我也要感谢队友在本次实验中的关怀和陪伴，让本次实验得以完满完成。