

计算机组成与体系结构（H） 实验报告

Lab8&9 【异常】



学生姓名： 陈乐屿

学 号： 22307130104

专 业： 计算机科学与技术

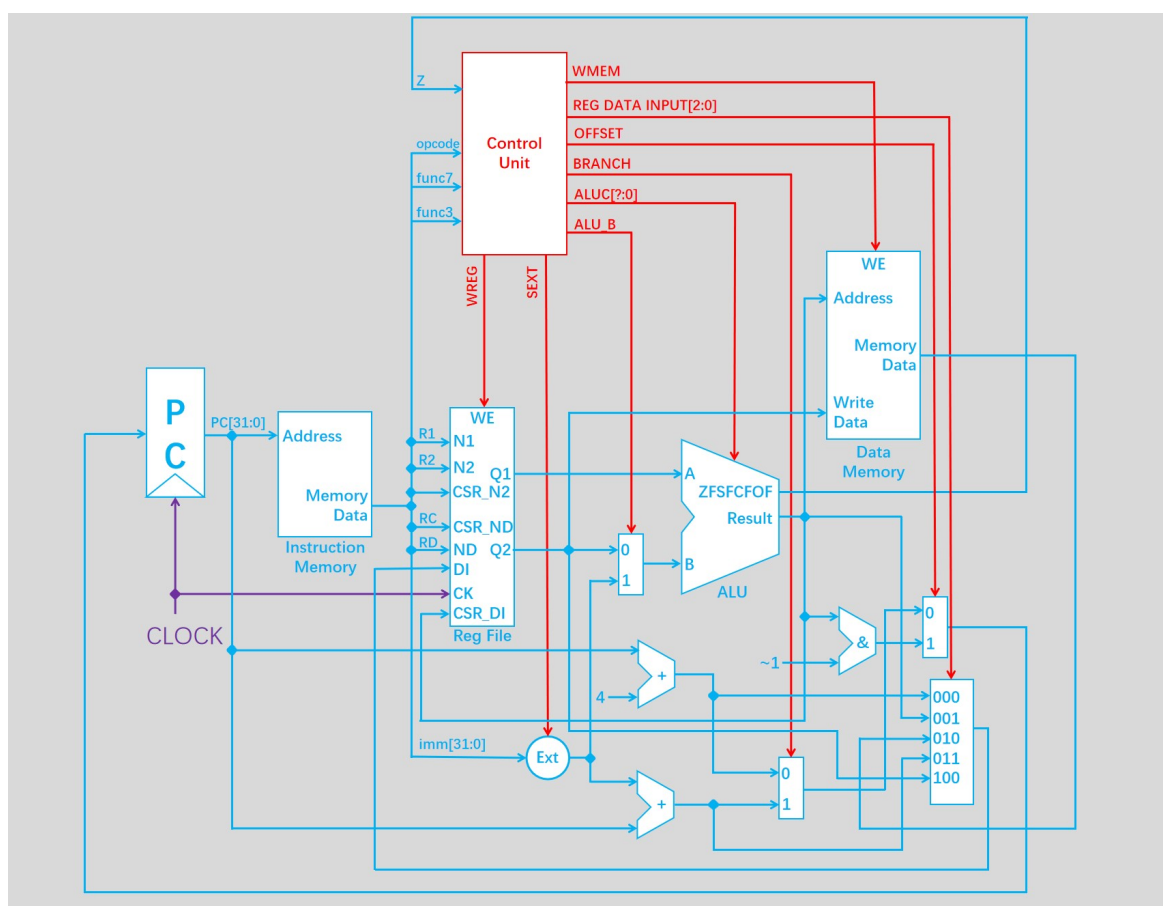
一、实验目的

在前7次实验的可实现64位RISC-V指令功能的5级流水线CPU的基础上，添加CSR与异常指令，以支持运行操作系统，并上板。

二、实验内容

2.1 电路图

根据新添加的CSR与异常指令，重新绘制CPU的电路图如下（因为是在单周期CUP的电路图的基础上进行修改，所以省略了各级流水线寄存器）：



2.2 寄存器模块

主要修改的地方是寄存器模块，包括其在F阶段、E阶段与W阶段的行为。下面以csrrc与mret指令为例。

csrrc rd, csr, rs1 $t = \text{CSRs}[\text{csr}]; \text{CSRs}[\text{csr}] = t \& \sim x[\text{rs1}]; x[\text{rd}] = t$
 读后清除控制状态寄存器 (Control and Status Register Read and Clear). I-type, RV32I and RV64I.
 记控制状态寄存器 *csr* 中的值为 *t*。把 *t* 和寄存器 *x[rs1]* 按位与的结果写入 *csr*，再把 *t* 写入 *x[rd]*。

31	20 19	15 14	12 11	7 6	0
csr	rs1	011	rd	1110011	

执行 mret 指令时，进行以下操作：

- `pc <- mepc`
- `mstatus.mie <- mstatus.mpie`
- `mstatus.mpie = 1'b1`
- `mstatus.mpp <- 2'b0`
- 清除流水线。取消当周期发起的 `dreq.valid`。已发起的 `dreq` 保留，等到 `data_ok` 后再清除流水线。

在F阶段，mret需要修改pc值，由于CSR相关指令操作时需要清空流水线，因此无需考虑控制冒险，让指令自然流入到E阶段计算新pc值即可。另外，每一条CSR相关指令都需要独占流水线，所以需要阻塞F阶段直到CSR相关指令被提交。

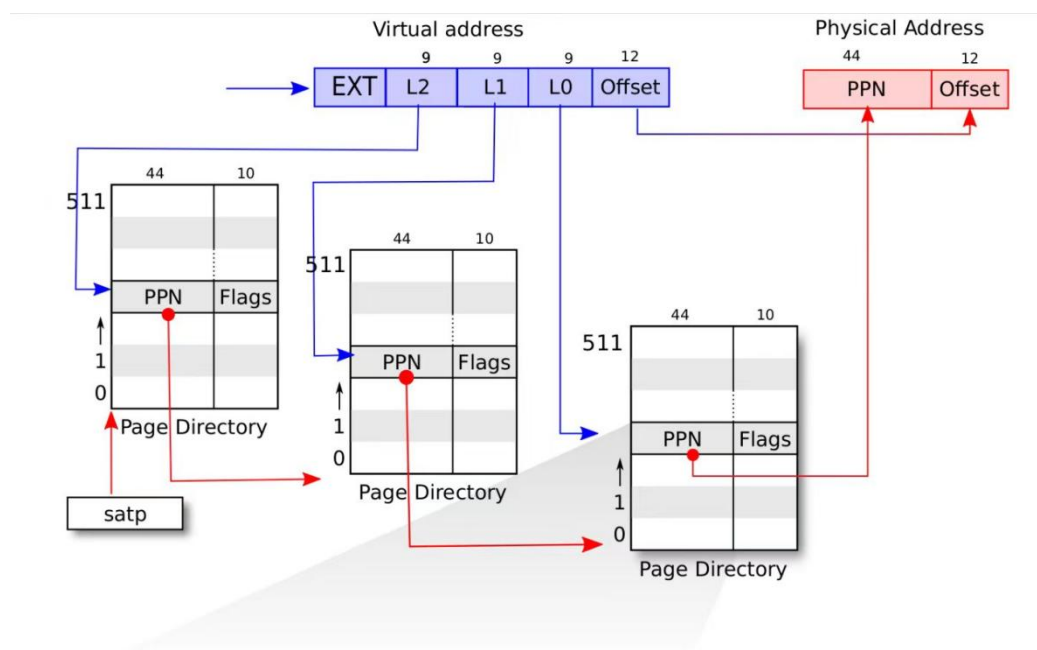
在E阶段，csrrc需要读取 $x[\text{rs1}]$ 与 $\text{CSRs}[\text{csr}]$ 的内容，并输入到ALU中进行计算；mret需要读取 $\text{CSRs}[\text{mepc}]$ 的内容，并用于更新PC值。

在W阶段，csrrc需要同时对 $x[\text{rd}]$ 与 $\text{CSRs}[\text{csr}]$ 进行写入；mret需要对 $\text{CSRs}[\text{mstatus}]$ 进行写入。因此，在寄存器模块上设置2个写入口，分别用于普通寄存器与CSR的写入。另外，对于mret和ecall两个指令，由于CSR相关指令操作时需要清空流水线，因此无需考虑数据冒险，又因这两个指令不需要进行运算，所以可以直接在寄存器模块内部进行CSR的数据转移。

2.3 访存模块

因为引入了虚拟地址，因此访存模块也需要进行一定的修改，包括其在F阶段与M阶段的行为。

在判断是否需要地址翻译时，需要同时检查mode与satp寄存器（并且不需要考虑数据冒险，方便了代码的设计）。当 $mode \neq 3$ 且 $satp.mode$ （即 $satp[63:60]$ ） $\neq 8$ 时，虚拟地址 \neq 物理地址，需要从虚拟地址和satp的内容出发，进行3次额外访存来获得物理地址。额外访存可以通过在状态机上简单增加若干分支状态来实现，当需要进行地址翻译时就依次进入这些状态。



除第一次访存外，每次访存需要用前一次访存得到的数据（页表项中存放的物理地址）计算出本次访存的物理地址。

由于需要用到64位的数据，且ibus只能返回32位的数据（因为指令只有32位），因此即使是F阶段也必须要用dbus得到指令的物理地址后再通过ibus得到指令数据。

在我原本的程序中，F阶段与M阶段分别写在两个always_ff块中，分别调用ibus与dbus。因为F阶段也需要调用dbus，所以dreq存在多驱动问题，因此需要把F阶段与M阶段合成写在同一个always_ff块中。

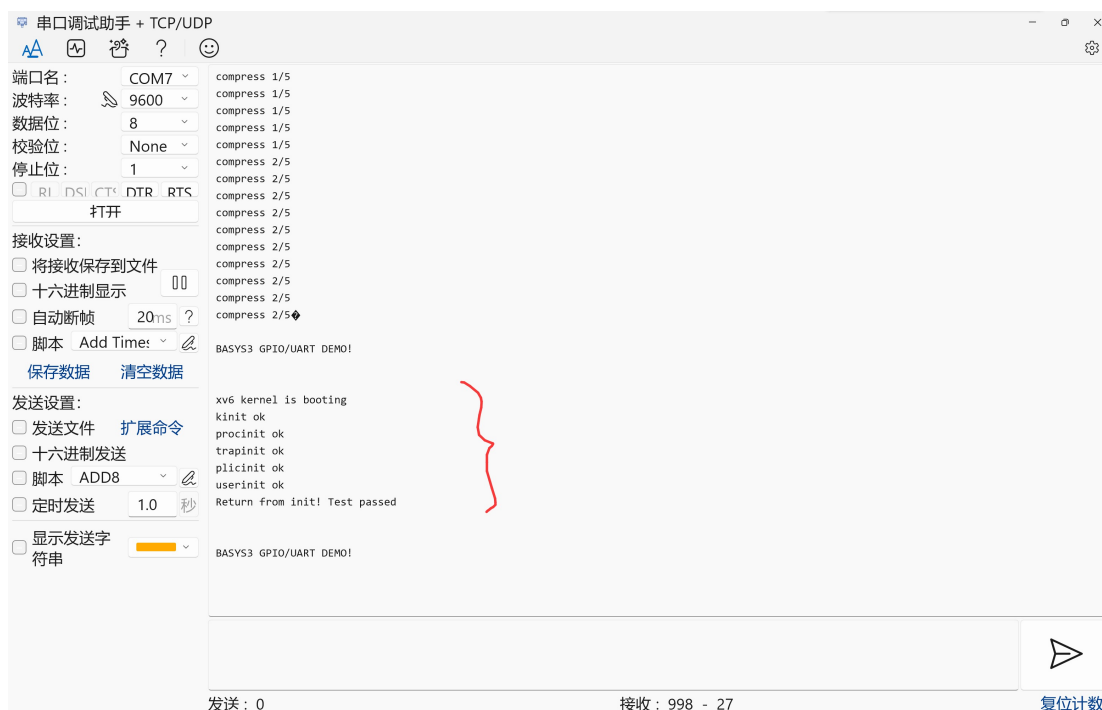
在原本的写法中，F与M阶段是2个状态机。在所有阶段都解除阻塞后，它们从空闲状态进入准备状态。M阶段会先判断是否拉起dbus请求，F阶段会等待M阶段结束dbus请求(不拉起请求，或dresp.ok==1)后再拉起ibus请求。当获取到数据并存储完毕后，就进入空闲状态。

在新的写法中，F与M阶段合成为1个状态机。所有阶段都解除阻塞后，从空闲状态进入M阶段的准备状态，M阶段的工作完成后进入F阶段，最后回到空闲状态。需要特别注意的是，F阶段存在气泡状态，因此需要设一个信号来记录：M阶段结束后，是进入F阶段的准备状态，还是进入F阶段的某个气泡状态。

三、仿真测试(verilator)

```
chenleyu@chenle x + v
ccache g++ -I. -MMD -I/usr/local/share/verilator/include -I/usr/local/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0 -DVM_TRACE=1 -DVM_TRACE_FST=1 -
faligned-new -fcf-protection=none -Wno-bool-operation -Wno-sign-compare -Wno-uninitialized -Wno-unused-but-set-variable -Wno-unused-parameter -Wno-unused-variab
le -Wno-shadow -g -std=c++11 -static -Wall -I/home/chenleyu/arch-2024/difftest/src/test/csrc -I/home/chenleyu/arch-2024/difftest/src/test/csrc/common -I/hom
e/chenleyu/arch-2024/difftest/src/test/csrc/difftest -DVERILATOR -Wno-maybe-uninitialized -DNUM_CORES=1 -std=gnu++14 -c -o VSimTop__Trace__Slow.o VSimTop__Tr
ace__Slow.cpp
echo "" > VSimTop__ALL.verilator.deplist.tmp
Archive ar -rcs VSimTop__ALL.a VSimTop.o VSimTop___024root.o VSimTop___024unit.o VSimTop__Dpi.o VSimTop__Trace.o VSimTop___024root__Slow.o VSimTop___024unit__Sl
ow.o VSimTop__Syms.o VSimTop__Trace__Slow.o
g++ SimJTAG.o axi4.o common.o compress.o device.o flash.o keyboard.o ram.o remote_bitbang.o sdcad.o uart.o vga.o difftest.o goldenmem.o interface.o nemuprox
y.o ref.o spikedsasm.o main.o emu.o snapshot.o verilated.o verilated_dpi.o verilated_fst_c.o VSimTop__ALL.a -lpthread -lsdl2 -ldl -lz -l -o /home/chenleyu/
arch-2024/build/emu
rm VSimTop__ALL.verilator.deplist.tmp
make[4]: Leaving directory '/home/chenleyu/arch-2024/build/emu-compile'
make[3]: Leaving directory '/home/chenleyu/arch-2024/difftest'
make[2]: Leaving directory '/home/chenleyu/arch-2024/difftest'
make[1]: Leaving directory '/home/chenleyu/arch-2024'
TEST= ./build/emu --diff /home/chenleyu/arch-2024/ready-to-run/riscv64-nemu-interpreter-so -i ./ready-to-run/test-os/kernel || true
Emu compiled at Jun 14 2024, 12:31:32
The image is ./ready-to-run/test-os/kernel
Using simulated 256MB RAM
Using /home/chenleyu/arch-2024/ready-to-run/riscv64-nemu-interpreter-so for difftest
[src/device/io/mmio.c:18,add_mmio_map] Add mmio map 'clint' at [0xa2000000, 0xa200ffff]
[src/device/io/mmio.c:18,add_mmio_map] Add mmio map 'uartlite' at [0x40600000, 0x4060000c]
[src/device/io/mmio.c:18,add_mmio_map] Add mmio map 'uartlite1' at [0x23333000, 0x2333300f]
The first instruction of core 0 has committed. Difftest enabled.
xv6 kernel is booting
[WARNING] difftest store queue overflow
kinit ok
procinit ok
trapinit ok
plicinit ok
userinit ok
Return from init! Test passed
```

四、上板测试



前面的输出是在顺便测试test-cpu

五、建议与期待

一个学期的lab总算是画上了圆满的句号。陈老师学期初所说“这个学期要在CPU上跑操作系统”，当初还觉得是天方夜谭，但是在老师与助教的一步步引导下，我们也总算是克服了各种困难，真的在自己手搓的简陋的CPU上跑通了testos，虽然过程很痛苦，但是最后完成的时候还是很有成就感的。感谢陈老师深入浅出又充满风趣的教学，感谢助教学长学姐们辛苦设计的lab与悉心的指导，这一个学期的学习令我收获颇丰！