

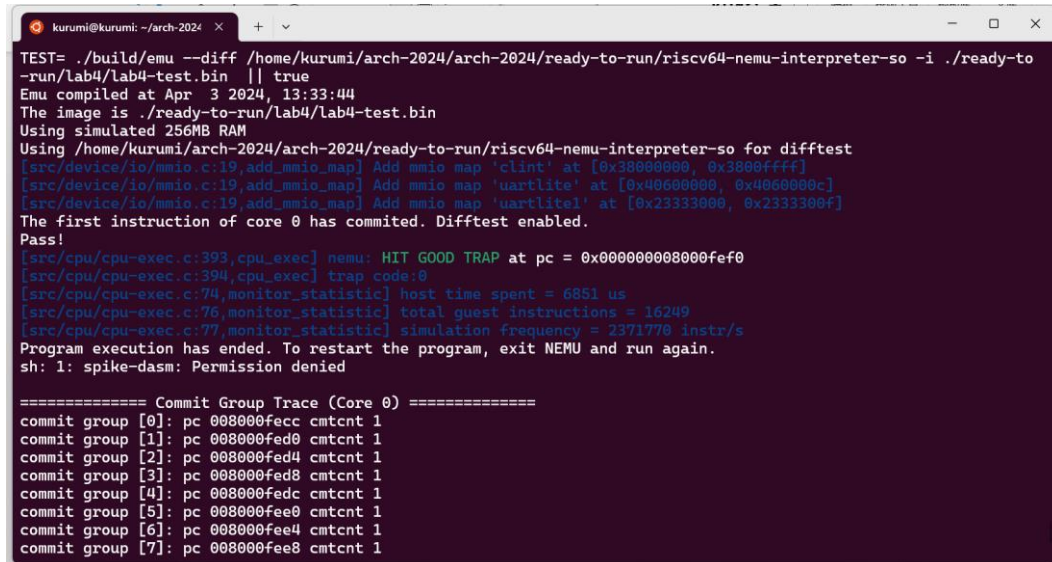
Lab4 实验

实验人：谢志康

学号：22307110187

时间：2024.4.3

通过截图：



```
kurumi@kurumi: ~/arch-2024
TEST= ./build/emu --diff /home/kurumi/arch-2024/arch-2024/ready-to-run/riscv64-nemu-interpretor-so -i ./ready-to-run/lab4/lab4-test.bin || true
Emu compiled at Apr  3 2024, 13:33:44
The image is ./ready-to-run/lab4/lab4-test.bin
Using simulated 256MB RAM
Using /home/kurumi/arch-2024/arch-2024/ready-to-run/riscv64-nemu-interpretor-so for diff test
[src/device/io/mmio.c:19,add_mmio_map] Add mmio map 'clint' at [0x38000000, 0x3800ffff]
[src/device/io/mmio.c:19,add_mmio_map] Add mmio map 'uartlite' at [0x40600000, 0x4060000c]
[src/device/io/mmio.c:19,add_mmio_map] Add mmio map 'uartlite1' at [0x23333000, 0x2333300f]
The first instruction of core 0 has committed. Diff test enabled.
Pass!
[src/cpu/cpu-exec.c:393,cpu_exec] nemu: HIT GOOD TRAP at pc = 0x000000000000fef0
[src/cpu/cpu-exec.c:394,cpu_exec] trap code:0
[src/cpu/cpu-exec.c:74,monitor_statistic] host time spent = 6851 us
[src/cpu/cpu-exec.c:76,monitor_statistic] total guest instructions = 16249
[src/cpu/cpu-exec.c:77,monitor_statistic] simulation frequency = 2371776 instr/s
Program execution has ended. To restart the program, exit NEMU and run again.
sh: 1: spike-dasm: Permission denied

===== Commit Group Trace (Core 0) =====
commit group [0]: pc 008000f0c cmtcnt 1
commit group [1]: pc 008000fed0 cmtcnt 1
commit group [2]: pc 008000fed4 cmtcnt 1
commit group [3]: pc 008000fed8 cmtcnt 1
commit group [4]: pc 008000fedc cmtcnt 1
commit group [5]: pc 008000fee0 cmtcnt 1
commit group [6]: pc 008000fee4 cmtcnt 1
commit group [7]: pc 008000fee8 cmtcnt 1
```

相较于上一版，添加了几条指令，大部分比较简单直接在 ALU 里新加几条判断即可。这一部分不讲了，就是按照指令手册一一完成即可，就是加了以下几行代码。

```
// lab4 add
`ALU_ADDW: begin TEMP = A + B; alu_c = {{32{TEMP[31]}}, TEMP[31:0]}; alu_f = 1'b0; end
`ALU_SUBW: begin TEMP = A - B; alu_c = {{32{TEMP[31]}}, TEMP[31:0]}; alu_f = 1'b0; end
`ALU_SLLW: begin TEMP = A << B[4:0]; alu_c = {{32{TEMP[31]}}, TEMP[31:0]}; alu_f = 1'b0; end
`ALU_SRLW: begin
    TEMP = {{32'b0}, A[31:0]} >> B[4:0];
    alu_c = {{32{TEMP[31]}}, TEMP[31:0]};
    alu_f = 1'b0;
end
`ALU_SRAW: begin
    T_SRAW_RES = ($signed(T_SRAW)) >>> B[4:0];
    alu_c = {{32{A[31]}}, T_SRAW_RES[31:0]};
    alu_f = 1'b0;
end

`ALU_ADDIW: begin TEMP = A + B; alu_c = {{32{TEMP[31]}}, TEMP[31:0]}; alu_f = 1'b0; end

// B[5:0] is shamt.
`ALU_SLLIW: begin TEMP = A << B[5:0]; alu_c = {{32{TEMP[31]}}, TEMP[31:0]}; alu_f = 1'b0; end
`ALU_SRLIW: begin
    TEMP = {{32'b0}, A[31:0]} >> B[5:0];
    alu_c = {{32{TEMP[31]}}, TEMP[31:0]};
    alu_f = 1'b0;
end
`ALU_SRAIW: begin
    T_SRAW_RES = ($signed(T_SRAW)) >>> B[5:0];
    alu_c = {{32{A[31]}}, T_SRAW_RES[31:0]};
    alu_f = 1'b0;
end
```

稍麻烦一点的部分是多周期乘除法器实现，但其实逻辑和我们 lab2 实现的访存指令差不多。在我的架构中，原本是有 stall, stall_this_dbus, stall_next_ibus, 用于控住 CPU，执行完当前指令再往前走。在 MEM 的访存指令中，我就是用以下逻辑实现：没有访存完（单周期搞不完）就让 stall_this_dbus 为 1（stall 就为 1）了，不让 cpu 读取下一条指令。这里的乘除法其实也就是这个逻辑。

具体乘除法器实现上网上很多教学，如何多周期算乘除，总思路就是每周期算一位（如参考https://blog.csdn.net/qq_44840079/article/details/104790338?spm=1001.2101.3001.6650.3&utm_medium=distribute.pc_relevant.none-task-blog-2~default~CTRLIST~default-3-104790338-blog-100743362.pc_relevant_antiscanv4&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2~default~CTRLIST~default-3-104790338-blog-100743362.pc_relevant_antiscanv4&utm_relevant_index=4）

我这里就以乘法器举例，64 位，我们分 64 个周期，每个周期算一位，之后移位，下一周期再算下一位（always_ff 实现，每个时钟上升沿计算一位，非阻塞赋值）最后得到结果。时钟控制的逻辑和 MEM 访存一样，我新开了个 stall_this_alu_mul，表示 mul 没算完将它控住（具体说来，也就是 mul 模块中有个 count，发现是 mul，count 开始设为 start，每周期算一位然后 count++，直到等于 64，算完，stall 恢复，cpu 才开始继续做事。

细节上也就是注意符号位并且特殊情况（除 0、溢出），特判解决即可。符号上我是开了个 sign 模块，先把俩操作数都变成正的，要是两者同号乘除运算后就是正，vice versa。所以之后都拿正的算（sign 的情况，unsign 拿原操作数），再看看是否 sign 使能为 1，是的话取反加一恢复即可。

最终完成。

（ps：之前在群里问有没有助教在 if 楼想去问问题是因为遇到了个冲突当时脑子没转过来不知道咋解决了，但其实很简单。就是因为，我原本有个 alu，乘除我新开了俩 alu 分别做事，然后最后该把谁的结果写到寄存器呢？当时没判断，我以为它自己会做事，但其实会冲突，显然，alu 没算乘除结果是 0，alu_mul 算乘法是一个结果，该把谁写回去，并不是自动把非零写回去，所以会有冲突。当时觉得写的很对了不知道为啥有一个使能一直为 0，看波形图发现没到除法那 64 个周期就开始了，百思不得其解。后面想清楚是这个问题，新加了一个 mux 模块决定最终 alu 计算结果就可以过了）