项目申请书

Created	@June 3, 2023
Tags	
weather	

项目名称：为SimpleO3Sim完善功能支持

项目主导老师：chenguokai

申请人：周鹏宇

日期：2023.6.3

邮箱：zhoupengyu19@mails.ucas.ac.cn

1.项目背景

1.项目基本需求

1.实现完整的 RV32IZicsr 指令集支持

目前项目模拟器已经支持了RISC-V的多数基本指令，并能够通过对应的标准测试；但几乎没有涉及到支持控制状态寄存器（Control and Status registers，即csr）及其指令的部分，这是本项目需要解决的第一个问题，即实现CSR以及支持CSR相关指令

2.实现该模拟器与其他行为级模拟器的行为对齐机制

不同的模拟器在同一套系统标准和输入下输出相同结果是用于检验本模拟器正确性的重要手段，同时也便于之后对该模拟器进行功能补充时进行正确性检验，是具有持久化价值的任务

3.整理测试用例形成单元测试

单元测试的目的在于用较少的测试开销检测尽可能多的模拟器功能，同样也是验证正确性的重要工作，且具有持久化价值

2.项目相关仓库

项目成果仓库： https://github.com/Simple-XX/SimpleO3Sim

2.技术方法及可行性

1.CSR相关

本人曾经系统的学习并实现过RISV-V和Loongarch CPU，且在基于Loongarch的CPU中实现过对CSR以及相关指令的支持，且阅读过对应文档，CSR指令的实现应当基于项目现有的各级部件的模拟器进行，而CSR本身的实现则只需要实现文档中提及的功能即可，需要注意的是一些特殊的读取/写入操作和特权级的判定。

本人的Loongarch CPU仓库：https://github.com/MerlinZou/2021_CA_lab

2.行为对齐和测试样例相关

目前针对RISC-V模拟器的成熟开源项目已经有很多，如：

- Spike，Spike是RISC-V基金会官方开发的RISC-V模拟器，它是可扩展和可定制的，可用于RISC-V体系结构的各个子集和扩展。Spike由C++编写，具有高度的可移植性，

- QEMU：QEMU是一种通用的模拟器，也支持RISC-V体系结构，也是本人接触最多也最熟悉的模拟器。QEMU支持多种操作系统和体系结构，包括RISC-V体系结构。QEMU可以与GDB调试器一起使用，还支持许多RISC-V扩展 https://github.com/qemu/qemu

此外还有诸如Renode，Ripes等，通过与上述模拟器进行对比，可以实现行为对齐的需求，此外针对RISC-V模拟器的测试样例也有大量的开源项目，通过对不同功能的样例进行组合测试，可以高效的生成针对特定功能的覆盖率高且任务开销较小的测试集。

3.项目实现的具体思路

1.CSR寄存器及指令实现

RISC-V有诸多CSR寄存器以及指令，如：

Number	Privilege	Name	Description
		User Trap Setup	
0x000	URW	ustatus	User status register.
0x004	URW	uie	User interrupt-enable register.
0x005	URW	utvec	User trap handler base address.
		User Trap Handling	
0x040	URW	uscratch	Scratch register for user trap handlers.
0x041	URW	uepc	User exception program counter.
0x042	URW	ucause	User trap cause.
0x043	URW	utval	User bad address or instruction.
0x044	URW	uip	User interrupt pending.
		User Floating-Point CSRs	
0x001	URW	fflags	Floating-Point Accrued Exceptions.
0x002	URW	frm	Floating-Point Dynamic Rounding Mode.
0x003	URW	fcsr	Floating-Point Control and Status Register (frm + fflags).
		User Counter/Timers	
0xC00	URO	cycle	Cycle counter for RDCYCLE instruction.
0xC01	URO	time	Timer for RDTIME instruction.
0xC02	URO	instret	Instructions-retired counter for RDINSTRET instruction.
0xC03	URO	hpmcounter3	Performance-monitoring counter.
0xC04	URO	hpmcounter4	Performance-monitoring counter.
		⋮	
0xC1F	URO	hpmcounter31	Performance-monitoring counter.
0xC80	URO	cycleh	Upper 32 bits of cycle, RV32I only.
0xC81	URO	timeh	Upper 32 bits of time, RV32I only.
0xC82	URO	instreth	Upper 32 bits of instret, RV32I only.
0xC83	URO	hpmcounter3h	Upper 32 bits of hpmcounter3, RV32I only.
0xC84	URO	hpmcounter4h	Upper 32 bits of hpmcounter4, RV32I only.
		⋮	
0xC9F	URO	hpmcounter31h	Upper 32 bits of hpmcounter31, RV32I only.

31 20	19 15	14 12	11 7	6 0
csr	rs1	funct3	rd	opcode
12	5	3	5	7
source/dest	source	CSRRW	dest	SYSTEM
source/dest	source	CSRRS	dest	SYSTEM
source/dest	source	CSRRC	dest	SYSTEM
source/dest	uimm[4:0]	CSRRWI	dest	SYSTEM
source/dest	uimm[4:0]	CSRRSI	dest	SYSTEM
source/dest	uimm[4:0]	CSRRCI	dest	SYSTEM

其实现的核心思路在于：

- 定义结构体，规定其大小，功能乃至于地址，掩码和初始值等
- 实现对寄存器的读写过程，其中对一些特殊的寄存器，如mtvec等，需要特殊实现其读写
- 对之前ID级等部分的功能进行补充，如decoding以及写后读问题等

2.行为对齐和测试样例相关

RISC-V的测试样例主要基于现有的开源仓库**riscv-tests**，https://github.com/riscv-software-src/riscv-tests，另外处于减少测试开销的考虑可以对部分用例进行组合测试。

行为对齐则采用软件工程领域的差分测试，选取QEMU模拟器作为金标准，设计部分API接口，如数据搬运信息，寄存器状态信息，执行指令数等，实现**在线指令级**验证。

4.规划

1.第一阶段（7.1-7.31）

- [ ] 实现CSR寄存器和指令集的设计
- [ ] 配置并通过RISC-V测试样例

2.第二阶段（8.1-8.31）

- [ ] 选取合适的现有开源RISC-V模拟器作为金标准，并设计API等diff测试需要的函数
- [ ] 实现对一个金标准模拟器的行为对齐

3.第三阶段（9.1-9.30）

- [ ] 尝试补充和丰富测试集，完成对之前工作的检验

期望

由于之前主要接触的是用硬件描述语言实现一个CPU，或用C等编程语言实现一个OS，希望本次项目能获取用C等语言对处理器进行模拟的全新经验。