# Sail Model简介

李威威
PLCT实验室

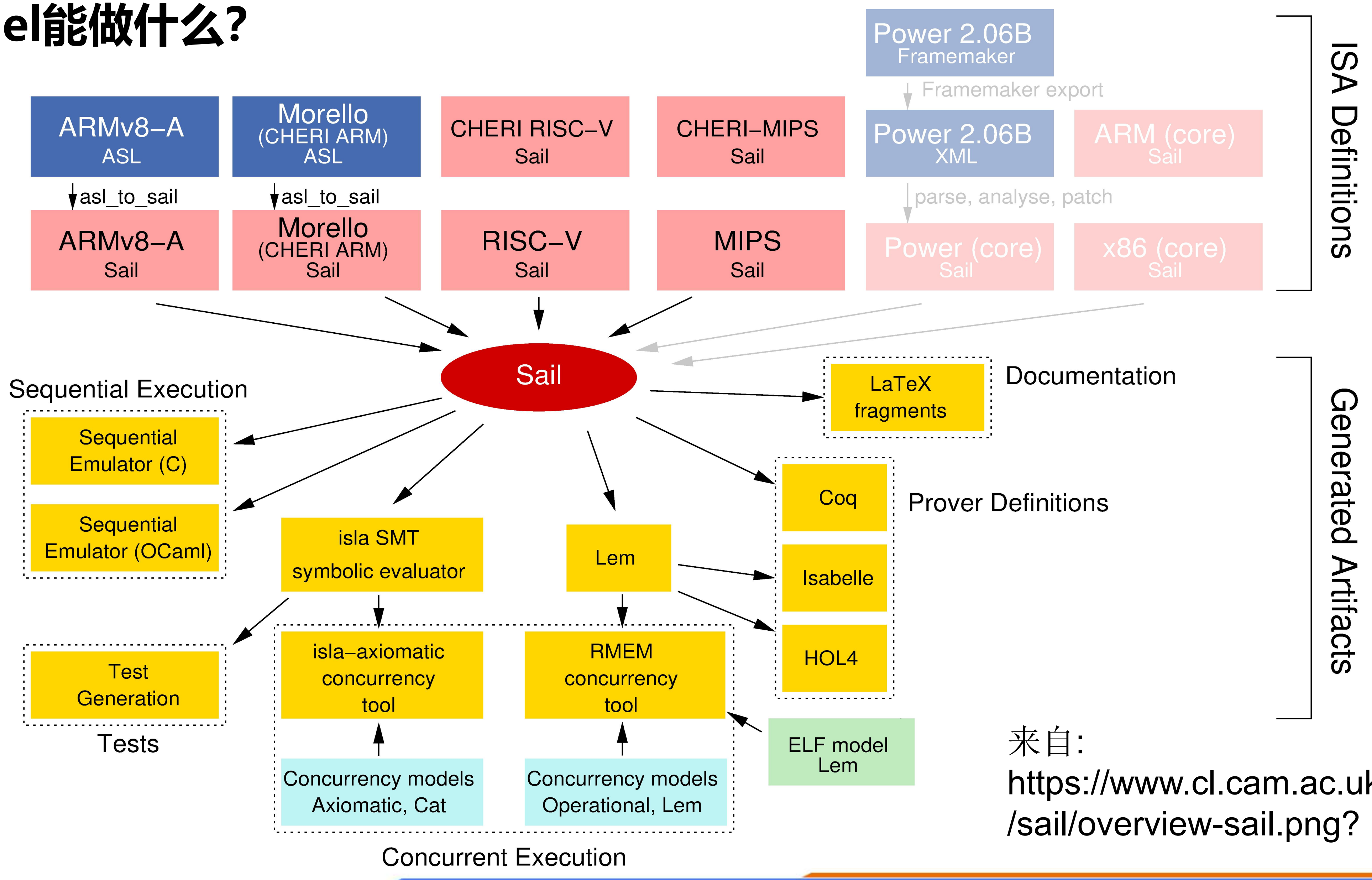
# 目 录

## Sail Model是什么?

**以Sail语言编写的针对指令集架构（ISA）的形式化模型**

- 指令的汇编语言格式
- 指令相应的编码器和解码器
- 指令的语义

## Sail Model能做什么?



来自:
https://www.cl.cam.ac.uk/~pes20/sail/overview-sail.png?

## RISC-V Sail Model简介

### 目前RISC-V Sail Model所支持的指令集架构：

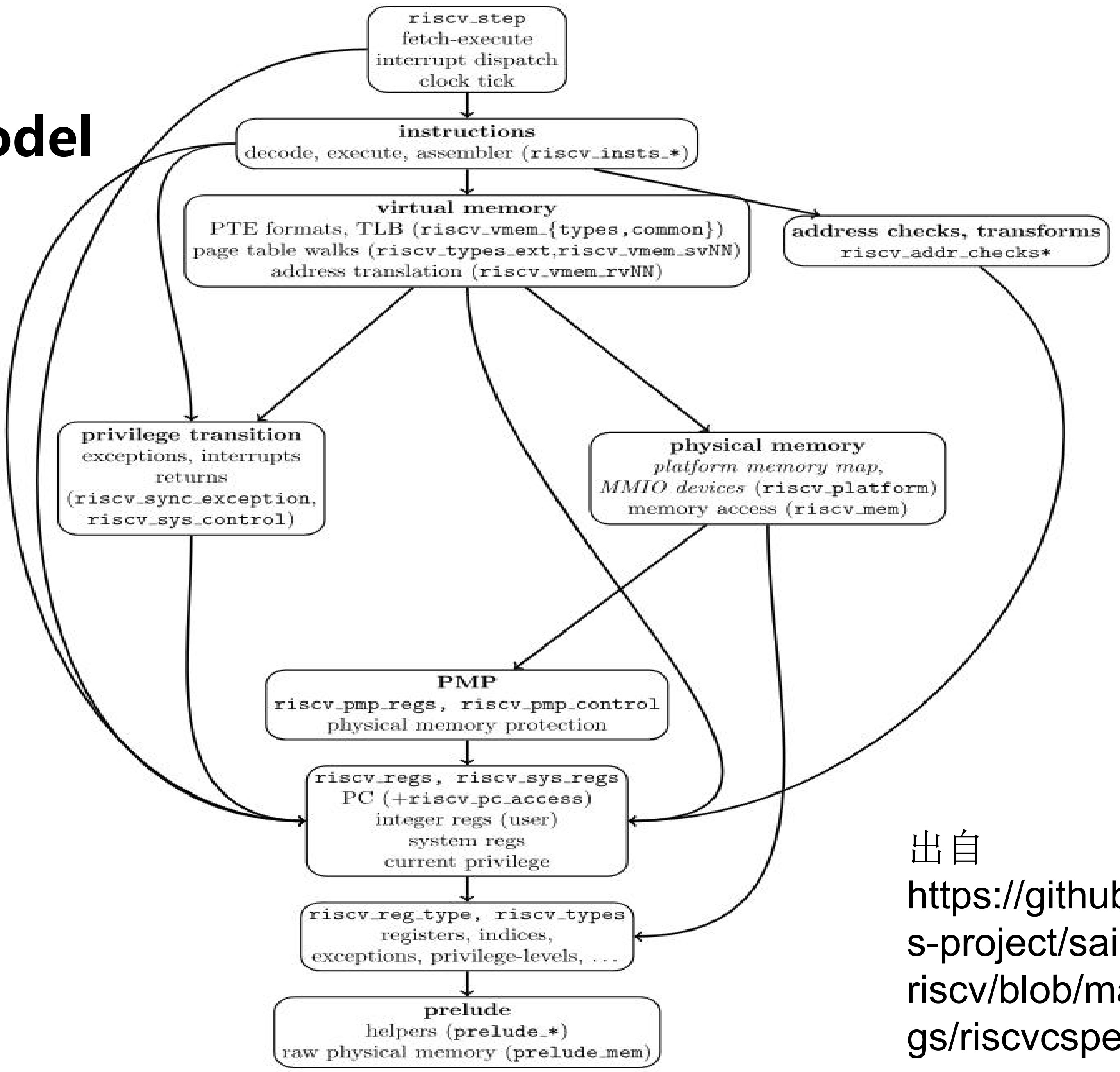
- RV32I/RV64I基础指令级
- M，A，C标准扩展
- F，D浮点标准扩展，只能在C仿真器中运行（使用SoftFloat C库）
- Zicsr控制和状态寄存器标准扩展
- N标准扩展（用户级中断）
- 基础计数器和计时器
- M和S-Mode RV32,RV64指令集架构
- PMP
- Zfinx, Zsn, Scalar K 0.8.1扩展 (PR...)

## RISC-V Sail Model代码结构

```
sail-riscv
- model                    // Sail specification modules
- generated_definitions    // files generated by Sail, in RV32 and RV64 subdirectories
  -  c
  -  ocaml
  -  lem
  -  isabelle
  -  coq
  -  hol4
  -  latex
- prover_snapshots         // snapshots of generated theorem prover definitions
- handwritten_support      // prover support files
- c_emulator               // supporting platform files for C emulator
- ocaml_emulator           // supporting platform files for OCaml emulator
- doc                      // documentation, including a reading guide
- test                     // test files
  - riscv-tests            // snapshot of tests from the riscv/riscv-tests github repo
- os-boot                  // information and sample files for booting OS images
```

## RISCV Sail Model

## 依赖图



riscv_step
fetch-execute
interrupt dispatch
clock tick
instructions
decode, execute, assembler (riscv_insts_*)
virtual memory
PTE formats, TLB (riscv_vmem_{types,common})
page table walks (riscv_types_ext,riscv_vmem_svNN)
address translation (riscv_vmem_rvNN)
address checks, transforms
riscv_addr_checks*
privilege transition
exceptions, interrupts
returns
(riscv_sync_exception,
riscv_sys_control)
physical memory
platform memory map,
MMIO devices (riscv_platform)
memory access (riscv_mem)
PMP
riscv_pmp_regs, riscv_pmp_control
physical memory protection
riscv_regs, riscv_sys_regs
PC (+riscv_pc_access)
integer regs (user)
system regs
current privilege
riscv_reg_type, riscv_types
registers, indices,
exceptions, privilege-levels, . . .
prelude
helpers (prelude_*)
raw physical memory (prelude_mem)

出自
https://github.com/rems-project/sail-riscv/blob/master/doc/figs/riscvcspecdeps.svg

## 指令模拟: riscv_step.sail

```
let (retired, stepped) : (Retired, bool) =
    match dispatchInterrupt(cur_privilege) {
      Some(intr, priv) => { ...},
      None() => {
        /* the extension hook interposes on the fetch result /
        let f : FetchResult = ext_fetch_hook(fetch());
        match f {
                / extension error /
                F_Ext_Error(e)    => { ...},
                / standard error /
                F_Error(e, addr) => {...},
                / non-error cases: */
                F_RVC(h) => {... },
                F_Base(w) => {
                                let ast = decode(w);
                                ...
                                nextPC = PC + 4;
                                (execute(ext_post_decode_hook(ast)), true)
                }
        }
      }
}
```

中国科学院软件研究所 Institute of Software Chinese Academy of Sciences | 智能软件研究中心 Intelligent Software Research Center

## 指令模拟: riscv_insns_*.sail

### 涉及的Sail文件:

- riscv_insts_begin.sail
- riscv_insts_end.sail
- riscv_insts_base.sail
- riscv_insts_aext.sail
- riscv_insts_mext.sail
- riscv_insts_dext.sail
- riscv_insts_next.sail
- riscv_insts_fext.sail
- riscv_insts_zicsr.sail
- riscv_insts_cext.sail, riscv_insts_cfext.sail,riscv_insts_cdext.sail
- riscv_insts_hints.sail
- *riscv_insts_rmem.sail*

## 指令模拟: riscv_insns_begin.sail

```
scattered union ast

/* returns whether an instruction was retired, used for computing minstret */
val execute : ast -> Retired effect {escape, wreg, rreg, wmv, wmvt, eamem, rmem, rmemt, barr, exmem, undef}
scattered function execute

val assembly : ast <-> string
scattered mapping assembly

val encdec : ast <-> bits(32) effect {rreg}
scattered mapping encdec

val encdec_compressed : ast <-> bits(16) effect {rreg}
scattered mapping encdec_compressed
```

## 指令模拟-指令编码: riscv_insns_base.sail

```
union clause ast = ITYPE : (bits(12), regidx, regidx, iop)

mapping encdec_iop : iop <-> bits(3) = {
        RISCV_ADDI <-> 0b000,
        RISCV_SLTI <-> 0b010,
        RISCV_SLTIU <-> 0b011,
        RISCV_ANDI <-> 0b111,
        RISCV_ORI <-> 0b110,
        RISCV_XORI <-> 0b100
}

mapping clause encdec = ITYPE(imm, rs1, rd, op)
<-> imm @ rs1 @ encdec_iop(op) @ rd @ 0b0010011
```

## 指令模拟-指令功能: riscv_insns_base.sail

```
function clause execute (ITYPE (imm, rs1, rd, op)) = {
let rs1_val = X(rs1);
let immext : xlenbits = EXTS(imm);
let result : xlenbits = match op {
        RISCV_ADDI => rs1_val + immext,
        RISCV_SLTI => EXTZ(bool_to_bits(rs1_val <_s immext)),
        RISCV_SLTIU => EXTZ(bool_to_bits(rs1_val <_u immext)),
        RISCV_ANDI => rs1_val & immext,
        RISCV_ORI => rs1_val | immext,
        RISCV_XORI => rs1_val ^ immext
};
X(rd) = result;
RETIRE_SUCCESS
}
```

## 指令模拟-指令汇编: riscv_insns_base.sail

```
mapping itype_mnemonic : iop <-> string = {
        RISCV_ADDI <-> "addi",
        RISCV_SLTI <-> "slti",
        RISCV_SLTIU <-> "sltiu",
        RISCV_XORI <-> "xori",
        RISCV_ORI <-> "ori",
        RISCV_ANDI <-> "andi"
}
mapping clause assembly = ITYPE(imm, rs1, rd, op)
<-> itype_mnemonic(op) ^ spc() ^ reg_name(rd) ^ sep() ^ reg_name(rs1) ^ sep() ^ hex_bits_12(imm)
```

## 指令模拟: riscv_insns_end.sail

```
/* End definitions */
end ast
end execute
end assembly
end encdec
end encdec_compressed

val print_insn : ast -> string
function print_insn insn = assembly(insn)

overload to_str = {print_insn}

val decode : bits(32) -> ast effect {rreg}
function decode bv = encdec(bv)

val decodeCompressed : bits(16) -> ast effect {rreg}
function decodeCompressed bv = encdec_compressed(bv)
```

## 参考

- Sail RISCV代码：https://github.com/rems-project/sail-riscv
- SAIL RISCV状态：https://github.com/rems-project/sail-riscv/blob/master/doc/Status.md
- 阅读指南：https://github.com/rems-project/sail-riscv/blob/master/doc/ReadingGuide.md
- 扩展指南：https://github.com/rems-project/sail-riscv/blob/master/doc/ExtendingGuide.md
- 在Isabelle/HOL中使用Sail：https://raw.githubusercontent.com/rems-project/sail/0.12/snapshots/isabelle/Manual.pdf
- Sail简介：https://www.cl.cam.ac.uk/~pes20/sail/
- SAIL语言手册：https://github.com/rems-project/sail/blob/sail2/manual.pdf

# 谢谢各位

欢迎提问、讨论、交流合作