Listings

1 RISC-V

RISC-V is an Instruction Set Architecture. Listing 1 demonstrates syntax highlighting for its GNU assembly language.

```
#include "riscv_test.h"
#include "../test_macros.h"

RVTEST_RV64M
RVTEST_CODE_BEGIN

.align 2

li TESTNUM, 1
# Set mstatus.MPP to User (0b00) & mstatus.MPV to 0b1
csrr t1, mstatus
li t2, 0xffffff7fffffe7ff # Reset
and t1, t1, t2
li t2, 0x0000008000000000 # Set
or t1, t1, t2
csrw mstatus, t1
# Load address of 'vucode' into the machine exception program counter
la t1, vucode
csrw mepc, t1
mret              # Go to VU-mode
unimp

vucode:
    RVTEST_PASS

RVTEST_CODE_END

.data

# Output data section.
RVTEST_DATA_BEGIN
        .align 3

result:
        .dword -1
RVTEST_DATA_END
```

Listing 1: RISC-V example

2 Sail

Sail is a domain specific language for expressing ISA semantics.

```
function raises_virtual_instr(csr : csreg, p : Privilege, v:
    ↪ Virtualization, isWrite : bool) -> bool =
  if v == V0 then false
  else match p {
    Machine => internal_error(__FILE__, __LINE__, "illegal privilege
        ↪ mode"),
    Supervisor => {
      let csr_rw = csr[11..10];
      let csr_p  = csr[9..8];

      if not(isWrite == true & csr_rw== 0b11) & (csr_p <=_u 0b10) /* HS
          ↪ -access allowed? */
      then match (csr_rw, csr_p) {
        (_, 0b10) => true,  /* Hypervisor or VS CSR accesses from VS-
            ↪ mode */
        (_, _)    => false,
      } else false
    },
    User => {
      let csr_rw = csr[11..10];
      let csr_p = csr[9..8];

      if not(isWrite == true & csr_rw == 0b11) & (csr_p <=_u 0b10) /*
          ↪ HS-access allowed? */
      then match (csr_rw, csr_p) {
        (_, 0b01) => true,  /* Supervisor CSR accesses from VU-mode */
        (_, 0b10) => true,  /* Hypervisor or VS CSR accesses from VU-
            ↪ mode */
        (_, _)    => false,
      } else false
    },
  }
```

Listing 2: Sail example

3 Rust

Rust is a general purpose language which focuses on a combination of safety and speed.

```rust
pub struct Counter<const ADDRESS: u32> {
    reg: &'static mut Counter_Registers,
}

impl <const A: u32> Counter<A> {
    /// Create a new counter with a fixed base address
    pub fn new() -> Self{
        Counter {
            reg: unsafe { &mut *(A as *mut Counter_Registers) },
        }
    }

    /// Get `status_reg``s value
    pub fn get_status_reg(&self) -> u32 {
        unsafe {
            let sr_p: *const u32 = &(self.reg.sr); // Take pointer to status register
            sr_p.read_volatile() // Read from pointer
        }
    }

    /// Get `control_reg``s value
    pub fn get_control_reg(&self) -> u32 {
        unsafe {
            let cr_p: *const u32 = &(self.reg.cr); // Take pointer to command register
            cr_p.read_volatile() // Read from pointer
        }
    }

    /// Get `value_reg``s value
    pub fn get_value(&self) -> u32 {
        unsafe {
            let value_p: *const u32 = &(self.reg.vr); // Take pointer to value register
            value_p.read_volatile() // Read from pointer
        }
    }

    /// Set `control_reg``s value
    pub fn set_control_reg(&mut self, value: u32) -> () {
        unsafe {
            let cr_p: *mut u32 = &mut (self.reg.cr); // Take mutable pointer to command register
            cr_p.write_volatile(value) // Write to pointer
        };
    }
}
```

Listing 3: Rust example