VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Shwetank Shekhar 22BEC1204 <shwetank.shekhar2022@vitstudent.ac.in>

Subject: Issue Report & Tool-chain Query - Phase 1, Week 1

1 message

Shwetank Shekhar 22BEC1204 <shwetank.shekhar2022@vitstudent.ac.in>

Wed, Jun 4, 2025 at 5:10 PM

To: Kunal Ghosh <contactvsd@vlsisystemdesign.com>

Dear Kunal Ghosh

I'm writing to report a critical toolchain issue blocking progress in Week 1. The provided RISC-V toolchain appears to lack Zicsr extension support, preventing compilation of essential software like OpenSBI and pk.

I am currently working through the initial setup and basic tasks. The toolchain I began with is `riscv-toolchain-rv32imac-x86_64-ubuntu.tar.gz`, as understood from the program materials.

Summary of Tasks and Issues:

1. **Task 1: Tool-chain Installation & Sanity Check:**
   - The `PATH` was set up correctly for `riscv32-unknown-elf-gcc` and `objdump`.
   - An initial issue with `riscv32-unknown-elf-gdb` failing due to a missing `libpython3.10.so.1.0` was resolved by installing the `libpython3.10-dev` package. All three core tools (`gcc`, `objdump`, `gdb`) were then verified.
2. **Task 2: Compile "Hello, RISC-V":**
   - My "Hello, RISC-V!" C program (`hello_riscv_shwetank.c`) failed to compile with the specified `-march=rv32imc` flag, giving the error: `riscv32-unknown-elf-gcc: fatal error: Cannot find suitable multilib set for '-march=rv32imc'/'-mabi=ilp32'`.
   - We found that `riscv32-unknown-elf-gcc --print-multi-lib` returned `.;`, indicating only a default multilib.
   - **Workaround:** The compilation succeeded using `-march=rv32imac -mabi=ilp32`. The resulting ELF file was verified.
3. **Tasks 3-5 (Assembly, Hex Dump, ABI Registers):** These were completed successfully using the `rv32imac` compilation flag.
4. **Task 6: Stepping with GDB (`target sim`):**
   - Attempts to use GDB with `target sim` for the compiled ELF were problematic.
   - Initial `run` commands (after `target sim` and `break main`) led to "No program loaded" or "Cannot access memory" errors for the breakpoint.
   - Explicitly using `load` and setting the breakpoint by address (`break *0x10162`) resulted in the program terminating immediately with `SIGILL, Illegal instruction` before even reaching the breakpoint at the `_start` address (`0x100e2`).
   - Further investigation showed GDB reporting "The program has no registers now" after `load`, and `set $pc` commands failed.
   - We concluded that `target sim` with the current toolchain/ELF was proving unreliable for execution.
5. **Task 7: Running Under an Emulator (QEMU & OpenSBI Build Attempts - leading to toolchain issue discovery):**
   - **QEMU:**
     - Installed `qemu-system-misc` (which provides `qemu-system-riscv32`).
     - Running the `hello_riscv_shwetank_t2_output.elf` with `qemu-system-riscv32 -nographic -kernel ...` failed with: `Unable to find the RISC-V BIOS "opensbi-riscv32-generic-fw_dynamic.bin"`.
     - Installing the `opensbi` package did not resolve this, as `dpkg -L opensbi` showed it only provided 64-bit firmware under `/usr/lib/riscv64-linux-gnu/`. Similarly, `/usr/share/qemu/` only contained a 64-bit OpenSBI file.
     - Using `-bios none` with QEMU allowed a simple test program (without `printf`) to run, but the original "Hello, RISC-V!" program (with `printf`) ran without console output (likely due to

unhandled syscalls).

- **Attempt to Build 32-bit OpenSBI from Source (to use with QEMU):**
  - **OpenSBI v1.4:** Build failed due to the linker (`riscv32-unknown-elf-ld.bfd`) not recognizing the `--exclude-libs` option.
  - **OpenSBI `master` branch:** Build failed with `Makefile:196: *** Your linker does not support creating PIEs, opensbi requires this.. Stop.`
  - A direct test confirmed that my toolchain's linker indeed reports `-pie not supported`.
- **Attempt to Build RISC-V Proxy Kernel (`pk`) for Spike:**
  - The `pk` build failed with multiple errors: `Error: unrecognized opcode 'csrr...', extension 'zicsr' required` (e.g., in `entry.S`, `usermem.c`).

Identification of Root Cause - Tool-chain Issue:

The consistent failures in building `pk` (and the insights from OpenSBI build issues) led us to test the toolchain's assembler directly. A minimal assembly file (`test_csr.S`) containing a standard `csrr t0, mhartid` instruction failed to assemble with the following error, using both `-march=rv32imac` and `-march=rv32i`:

```
test_csr.S: Assembler messages:
test_csr.S:4: Error: unrecognized opcode `csrr t0,mhartid', extension `zicsr' required
```

This indicates that the `riscv32-unknown-elf-gcc` toolchain I am using (from `riscv-toolchain-rv32imac-x86_64-ubuntu.tar.gz`, GCC 14.2.0 based) has a fundamental problem: its assembler does not correctly recognize or support standard CSR instructions, which are a mandatory part of the base `rv32i` specification (via the `Zicsr` extension).

Current Blocker:

This toolchain issue prevents the successful compilation of essential support software like the RISC-V Proxy Kernel (`pk`) and OpenSBI. Without these, or a working alternative, I am unable to complete Task 7 (running the `printf`-enabled ELF in an emulator like Spike or QEMU with full functionality) and will likely face significant issues with subsequent development tasks that require standard ISA features.

Request for Guidance:

Could you please advise on the following:

1. Is the `riscv-toolchain-rv32imac-x86_64-ubuntu.tar.gz` I've been using the officially recommended tool-chain for this program?
2. If so, are there any known issues or additional configuration steps required for it, particularly regarding `Zicsr` support for `rv32imac` / `rv32i` targets?
3. If this toolchain is not suitable or known to be problematic, could you please point me to a stable, known-good `riscv32-unknown-elf` toolchain that is confirmed to work for the program's objectives, including building support software like `pk`?

I am eager to resolve this toolchain issue so I can proceed effectively with the program tasks.

Thank you for your time and assistance.

Sincerely,

Shwetank Shekhar