Below are the notes (unedited) from the various breakout sessions held at the end of Day 1 of the Workshop. These Breakouts may form the basis of the initial RISC-V Foundation Technical and Marketing Committees and sub-groups.

Platform Standardization **Jon Masters, RedHat; Arun Thomas, BAE Systems**

- Would like specs for different environments (microcontroller, server, etc.)
- Don't want to over specify
- Discuss bus interfaces
- In some cases, we don't want AXI
- Describe metadata common to all bus interfaces
- Focused on building generic software platforms
- Base platform that describes hardware expectations
- Want to avoid the situation ARM ran in to where you needed a different kernel for every SoC
- Hardware requirements (IOMMU, etc.) and software requirements (e.g., ABIs)
- Don: propose splitting the group in to those interested in an application core and those interested in microcontroller-class specs
- Farzad: There should be a working group established for RISC-V to look in to platform standardization
- Darius: What's the platform for RV32I Machine mode only
- Jon: ARM is good, but they didn't everything right. E.g., some GIC decisions
- Alex suggested BERI PIC as a starting point for the interrupt controller
- Possibly multiple interrupt controllers? One for microcontrollers, one for application processors
- Darius: Is there a way to determine which interrupt controller the system has? Feedback is it can be more difficult than you'd hope to switch different interrupt controllers in the Linux or FreeBSD kernels based on devicetree
- Maybe we should do some platform work as de facto (without the RISC-V stamp of approval) until we figure out the model
- We need a clean QEMU implementation of the platform
- ARM mach-virt may be a good model: virtio (serial, nic, disk), GIC, architected timers
- Could describe device configuration via DeviceTree or ACPI
- Upstreaming Linux kernel (experimental arch support)
- Plan to use mailing list for ongoing discussion, perhaps periodic meeting
- Hope to work with QEMU people on specifying a mach-virt style spec

Debug **Tim Newsome, SiFive; Joel Sandgathe, MicroVision**

- It is ...fairly... clear what everyone envisions & desires from a high level, a standalone chip that you can connect to via JTAG and exercise.
- How does a debug module inside RISCV plug in?
- How might it relate to "untethered lowRISC"?

- How might it relate to the way Rocket in Zynq ZC706 is accessed via some sort of supervisory interface by the Zynq's ARM core?
- Richard H has already set up OpenCores "advanced debug sys" and RiscV.
- There are features that relate more to HW verification and debug, vs. standard SW development.
- There should be mandatory and optional features.
- (SW) Typical breakpoints, halting, single-stepping,
    - register dump, memory dump,
    - instruction insertion,
    - jump to arbitrary location (and return and halt again)
- (HW) Can we halt processor and read out sufficient state information to reconstruct state in an emulator or model?
- (SW Trace): Replicate ARM ETM?
- (HW Trace): Full internal logic analyzer that could be pointed to various busses or parts of the pipeline? (available nodes would be implementation-dependent)
- What are the issues with the current proposal?
- Full CSR definition
    - Should there be a CSR to define instruction to insert, or address to jump to for arbitrary instructions?
- Self-identification of supported debug features by CPU
    - Device tree etc.
- Should there be an interface that is not the system bus?
    - Do we care about a situation where we have an implementation with no (or minimal) bus?
    - CSR control signals could be directly driven by debug interface
    - Registers could be dumped out directly on debug interface
    - We would like to define it in such a way that we can have either or both system bus interface and dedicated debug interface
- Issues:
    - It seems likely that we need to define a full halt rather than trap to debug RAM and loop.
    - There may be caching and privilege issues associated with putting debug ROM & RAM on system bus.
    - The current proposal of "everything achieved with debug RAM instructions" might not support use case of dumping all registers all the time.

Validation / Compliance Suite **Rishiyur Nikhil, Bluespec**

- Clarified the scope of this discussion:
    - not including static formal verification (proofs), which are interesting, but in the future
    - focusing on execution-based verification

- We discussed our experiences, and tried to draw some observations
- We each are doing some kind of A/B testing, comparing an implementation with a reference model.
- Kinds of test stimulus:
  - Very small hand-written code snippets
  - Randomly generated instruction sequences (most of which may be "meaningless", but the CPU still has to handle anything thrown at it).
  - Collected corpuses (corpi?) of application programs.
  - Full system code and loads
- But the specific details are all over the map
  - Each organization seems to rebuild this infrastructure internally, from scratch.
  - It often grows on demand, not in any planned way.
  - Even within an organization, sometimes this effort is duplicated in different parts of the organization.
  - They're often proprietary.
- Would be good to have an open-source, shared, reusable repository that has both:
  - The stimulus corpus (described above)
  - The testing infrastructure to conveniently plug in A/B components (model + implementation, model + new model, ...)

Marketing / Outreach **Peter Ateshian, NPS; Michael Gielda, Antmicro**

- biggest chance for RISC-V is little proprietary cores that only control things
- first complete implementation is in a dental camera ;)
- many people have asked about RISC-V in the US for the last 4 months
- Jothy from Draper: "same thing that happened to OSs, same is happening to instruction set"
- significant press mentions over coming months - Newsweek article
- besides standards, the first priority for the board should be marketing
- beg Feb - launch of new website not only gathering resources but also telling a story
- Peter Ateshian (MPS): what is the value proposition?
  - Common base for Electronic Design Processes
  - solving a problem existing since 1945, problems von Neumann architecture left us with
  - modern, rid of bloat, security built in: backporting security is hard
  - scalable, separates architecture from microarchitecture, defines only architecture
  - education, openness aspect, course materials are important
  - royalty-free ISA specification
  - allows benchmarking
  - things will win or lose based on their own merit and not lock-in
- we need to care about terminology and feed it back to journalists

- MicroSemi is interested in doing a low-cost board - ties into the RISC-V University Program, student contest, maybe even maker movement
- defense industry needs a trusted processor

QEMU Enablement **Sagar Karandikar, UC Berkeley**

- QEMU status - 80% of the way to being updated, should be done in a couple of weeks. done = 'boot linux and run software'
    - at this point probably just easier to finish then to have someone learn QEMU internals (unless someone else is an expert?)
- What can the community contribute?
    - Current reference to QEMU is "system mode" which acts like a piece of hardware. QEMU also has user mode, which is a TODO
        - However, proxy kernel on top of QEMU system mode works similarly
    - TCG backend for RISC-V
        - tcg is QEMU's internal representation:
            - RISC-V instructions -> tcg -> target machine code (e.g. x86)
            - simulating RISC-V on x86 is what we have now - RISC-V -> tcg
            - Future: tcg->RISC-V, so can run other ISA code on real RISC-V machines once they appear
- Upstreaming?
    - QEMU is GPL'd, but should be much easier than other software, QEMU RISC-V port contributor count is small
- Question about how to bring up new simulators easily:
    - make your simulator produce output that matches the spike debug output, diff

Memory Model / Formal Spec **Krste Asanovic, SiFive; Paolo Faraboschi, HPE**

- **Memory Model - Problem Intro**
    - Difficult to separate memory models from ISA
      memory models are all about ordering and visibility
    - Need to reconcile the needs of 3 different communities: hw architectures, formal folks, sw folks. Historically they don't understand each other. Efficiency tradeoffs are a big issue. Example: relaxed memory models cause fences everywhere, having often the opposite effect of what's desired
    - Overall principle: all memory operations are relaxed, ordering imposed by fences
    - Two separate problems
        - What is the formalism you want to use
        - What would you like the memory model to be
    - Would be nice to decouple the notion of what is the programmer's model from the mechanisms

    - Need to be able to capture complex memory systems containing a variety of elements, including NV memory, disaggregated memory, memory-side (or system-side) caching, multi-level main memories, etc.
    - What language bindings can we use to express ordering at these different points in the memory system?
- **Ordering primitives (eg, fences)**
    - Do we need to make fences more fine-grained?
    - For example, CELL had a bit mask to represent a range
    - By range? Could be difficult to express
    - By tag? Could be interesting because it is compatible with object consistency
    - Alternatively, we could use some form of a "DMA" operation (copy semantics) to implement "persistence"
    - Do we want to support some form of "relaxed fence"? OpenPower has some form of relaxed consistency atomics. Not clear what is the impact on the programming model. However, we need to make sure we don't mix up QoS issues with memory ordering issues.
- **Memory Errors, RAS, management mode**
    - Management and Machine mode
        - Should we consider memory models in management/machine mode as well? Largely independent from ordering, but there may be some issue around physical memory attributes and interaction with supervisor (hypervisor mode), whether the region can do atomics, cacheable, etc.
    - Memory errors and RAS
        - How to handle errors and propagate poison - poison on writes?
        - Probably requires a separate working group, maybe part of the platform standardization group
        - Do we need to report things "precisely"? Need to distinguish between software bugs (eg, write to a device that doesn't exist), from hw errors. This can be a great help to debug the software
        - Biggest impact is in platform profiles, to define how/what errors are caught and communicated
        - Need to leverage the work already done in x86 and ARM
- **Formal verification**
    - 4 types of ld/st (normal operation, virtual memory, hypervisor, I/O) need to understand how to represent all the 4 types in the formal specs

- Need to look at all the combinations of fences and dirty bits and make sure no holes are there
- How do we handle "legacy code", patterns like "store, store" sequence to implement a fence? Should we outlaw them?

Security Extensions **Silviu Chiricescu, BAE Systems; Dom Rizzo, Google**

- Security breakout recommendations:
    1. Security is a deep topic that touches many different domains; recommend continuing the discussion at Wednesday's breakouts.
    2. Recommend the foundation form a committee tasked with considering security and RISC-V
        - Security tends to be a broad concern, but there was consensus on the utility of a cryptographic accelerator ISA extension
    3. Recommend the security committee considers security instruction extensions to the ISA as its first topic
        - IP unencumbered side-channel resistant algorithms and implementations are a ripe area for research
        - The topic touches on secure boot, policy enforcement, debug, and many other areas
    4. Additional important topics to be considered by the security committee
        - Platform security concerns
            - Secure and/or verified boot
            - Secure enclaves (SGX, TrustZone, TPM, etc.)
            - Bus permission bits
        - Security policy enforcement
            - ex. CFI, Memory safety, Tagging
        - Debug interface security
            - Typically, a major vulnerability
            - Recommend that there be a representative from the security committee involved with debug

Notes:

- What kind of security are we considering?
    - Tagged architectures
    - Cryptography
    - Comprehensive security solution
- What's the taxonomy?
    - Side-channel analysis
    - Cryptographic acceleration and primitives; crypto extension
    - System management mode
    - Capability-based architectures -> formal reasoning
    - Boot-time security

    - Anti-malware(?)
        - Hard to write malware
    - Better memory management
    - Ring-model
    - IOMMU; IO-DMA protection
    - Control flow protection, memory protection
    - Tamper sensors and responses
    - SGX-like mode
    - TPMs
    - Design verification, implementation verification
    - Attestation
- Tagging (Policy enforcement mechanisms)
    - Standardize the way in which people think about tagging
    - How do you set tags; CSR, policy-enforcement
    - Need to have the ability to set and read tags
    - There's a taxonomy under tagging; are the meaning of tags determined at creation?
    - This is a big area
    - Is there room for a shared system? A single spec?
- Particular resonance w/ RISC-V
    - Open architectures ->
    - Andre -> can do tags w/out adding any new instructions
    - Enabled by extension specific CSRs
- Example tagged architectures
    - lowRISC
    - CHERI/BERI
    - MPX -> Hardwired policies
- Crypto accel extension?
    - Andre: many models of what is encrypted
        - anything in my secondary storage is encrypted
        - Anything out to DRAM gets encrypted
        - How do we put accelerators in there to be able to take advantage?
- Just secure boot; one signature
    - Networking and distributed permissions; lots of key management
    - Secure storage question
- 3 distinct questions
    - Should there be a crypto extension at all?
    - Should it be focused on userspace applications?
    - Or should it be a mode we enable?
- Generally supported instructions
    - More specialized operations

    - Specific curves, specific algorithms
- May not effect the ISA per se; but how does it fit into the platform
    - Consider networking and the disk
    - What would the stream-to-network crypto interface look like?
- Assume most SoCs have hardwired crypto accelerators
    - People will do crypto themselves
    - Would be worthwhile for the Foundation to do it?
- Secure enclaves
    - In TrustZone, security levels are exposed to the bus interface
        - Can expose the security state; there's value there
        - AXI bus includes defined privilege levels
- Side channel resistance
    - Concern at the micro-architectural level, algorithmic level, other levels?
        - Tend to find side channel attacks whenever we examine an area in detail; network timing, cache patterns, acoustic, EM, ...
    - IP concerns
        - If the group were to target a particular problem, could get traction and make progress.
    - ASCENT has tackled some of it
    - ORAM
    - Architects are generally not looking into this area, except:
        - UCSB
        - Dave Wagner
    - Is there academic work to be done here that could be used to push RISC-V extensions into commercial implementation?
        - Could be an interesting topic to pursue
        - Not much out in the open (IP unencumbered)
    - Overlaps strongly with cryptography extensions

Misc:

- Anything in the ISA that inhibits security choices?
    - Opcodes
- Anything in the ISA that aids security implementation?
    - Space in the CSRs
- Likely multiple security groups; w/ some overlap between them
    - Get the right people together
- NVIDIA developing an in-house processor w/ crypto accel
    - How do we switch b/w dfnt privilege modes?
    - Will they have the same switching process in both firmware and software?
    - Trusted boot ROM
    - If define it in the ISA, switching modes will be consistent
    - Something in the ISA; a secure world switch trap?

- Many security concerns are actually platform issues; ex. boot, bus permissions