

Unorthodox Architectures

Emad Jacob Maroun, PostDoc, ejama@dtu.dk

Unorthodox Instruction-Set Architectures (ISAs): ISA design has not changed much since the 70's. However, the design of processors and the needs of software have evolved immensely. Therefore, rethinking how ISAs are designed for the modern hardware and software landscape is necessary. **Scry** is a new ISA that gathers all new ISA design techniques into a cohesive whole not dependent on existing ISAs. It rethinks how data is tracked, how data flows between instructions, and accounts for the effects of caches on execution.

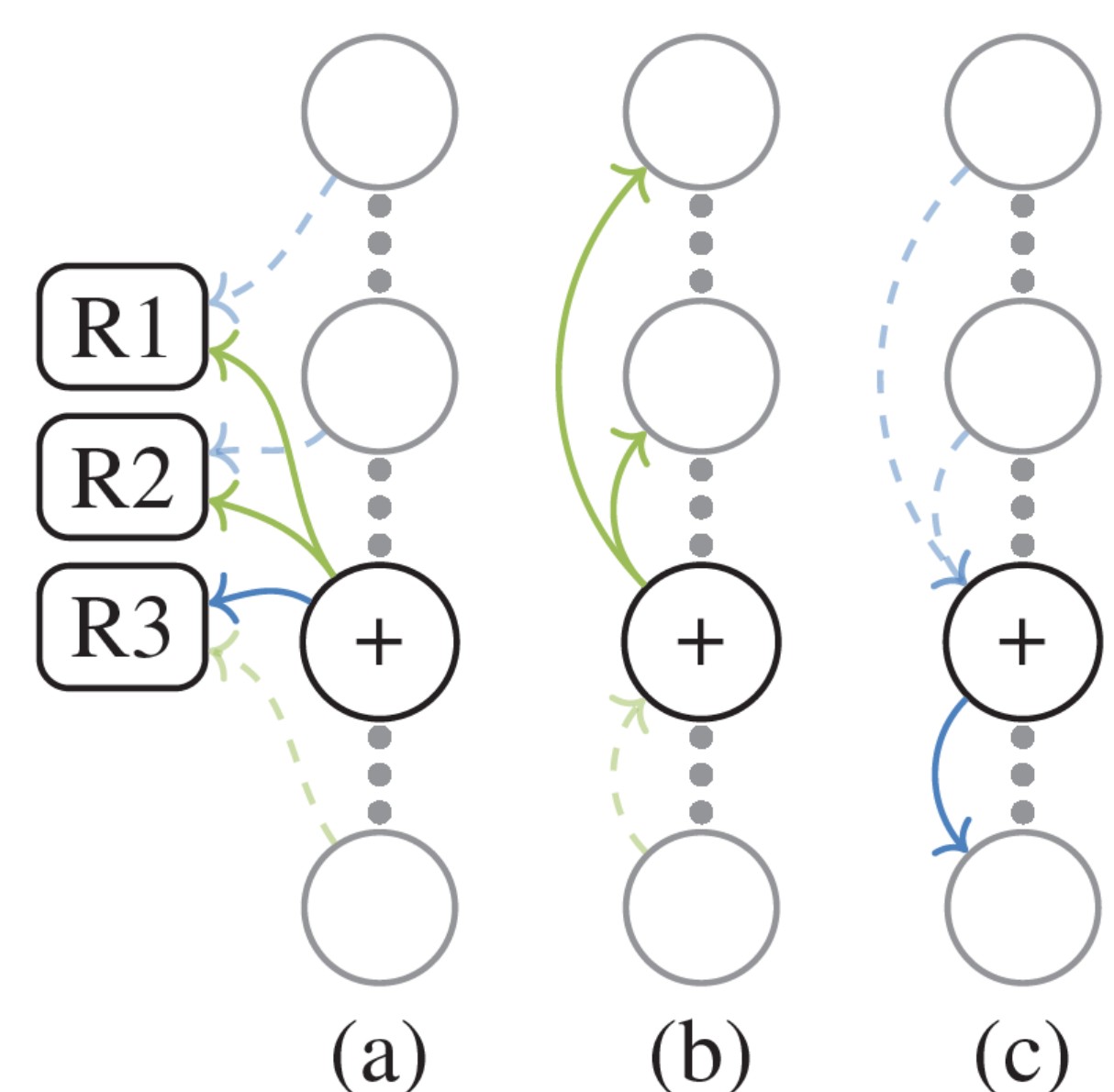
X-RISC-V Projects: Investigating the effects of novel ISA design techniques can benefit from individual technique research. Therefore, it would be interesting to design new ISAs based on RISC-V that incorporate new ideas like FTR or internal tagging. **Projects can include developing a tagged-RISC-V ISA to investigate the effects of tagging, or a FTR-RISC-V ISA for investigating the effects of FTR on more traditional ISAs.**

Operating System Projects: The Scry ISA can have significant effects on how operating systems should be built. Projects include **building a new operating system from scratch for the Nexys-A7 based CPU implementation.**

<code>addi</code>	<code>← addiw</code>	<code>add</code>	<code>← addw fadd.d fadd.s</code>
<code>c.add</code>	<code>← c.addw</code>	<code>sub</code>	<code>← subw fsub.d fsub.s</code>
<code>c.addi</code>	<code>← c.addiw</code>	<code>mul</code>	<code>← mulw fmul.s fmul.d</code>
<code>c.sub</code>	<code>← c.subw</code>	<code>mulh</code>	<code>← mulh mulhsu mulhu</code>
<code>c.srli</code>	<code>← c.srai</code>	<code>rem</code>	<code>← remu remw remuw</code>
<code>sll</code>	<code>← sllw</code>	<code>c.sw</code>	<code>← c.sd c.fsw c.fsd</code>
<code>slli</code>	<code>← slliw</code>	<code>c.swsp</code>	<code>← c.fswsp c.sdsp</code>
<code>blt</code>	<code>← bltu</code>	<code>srl</code>	<code>← sra srlw sraw</code>
<code>bge</code>	<code>← bgeu</code>	<code>srli</code>	<code>← srai srliw sraiw</code>
<code>slti</code>	<code>← sltiu</code>	<code>slt</code>	<code>← sltu flt.s flt.d</code>
<code>fcvt.d.w</code>	<code>← fcvt.s.w</code>	<code>fcvt.d.l</code>	<code>← fcvt.s.l</code>
<code>fcvt.d.wu</code>	<code>← fcvt.s.wu</code>	<code>fcvt.d.lu</code>	<code>← fcvt.s.lu</code>

Internal Tagging: Traditional ISAs do not track the type of values instead using typed instructions where needed (i.e. *add* vs *fadd* for integer or float addition). Internal tagging tracks the data's type and uses polymorphic instructions to reduce the number of instruction in the ISA and increase their flexibility.

Scry Projects: Scry is a clean-slate ISA design with unorthodox features: It uses data-type tagging so the processor always knows the type of a value, forward-temporal referencing (FTR) for data flow between instructions without using architectural registers, and deferred branches to reduce pipeline and cache-miss hazards for instruction fetching. Projects include **implementing a Scry CPU, compiler optimizations that enable efficient use of the new features, a LLVM-IR to Cranelift transpiler to allow for C/C++ compiling, or various developer tooling to help with assembly programming.**



Forward-Temporal Referencing: References between instructions. (a) Shows register references to both inputs (green) and the output (blue), (b) shows backward-temporal references to both inputs, and (c) shows forward-temporal references to only the single output.