COMPUTER SCIENCE

Motivation

Increasing usage of RISC-V at undergraduate level

RESEARCH-ARTICLE	RISC-V Console: A Containerized RISC-V Based Game Console Emulator for Education									
OPEN ACCESS July 2022	Christopher Nitta, Aaron Kaloti, Shuotong Wang									
\$	ITICSE '22: Proceedings of the 27th ACM Conference on on Innovation and Technology in Comp Education Vol. 1 • July 2022, pp 145–150 • https://doi.org/10.1145/3502718.3524791									
RESEARCH-ARTICLE OPEN ACCESS March 2021	RISC-V Reward: Building Out-of-Order Processors in a Computer Architecture Design with an Open-Source ISA									
\$	Stephen A. Zekany, S Jielun Tan, James A. Connelly, Ronald G. Dreslinski									
	SIGCSE '21: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education •									
ABSTRACT May 2019	Evolution and Revolution of Computer Systems Courses with the Open RISC-V ISA									
	Ke Zhang M Davida. PATTERSON									
	CompEd '19: Proceedings of the ACM Conference on Global Computing Education • May 2019, pp									
	171 • https://doi.org/10.1145/3300115.3312506									

Vast RISC-V ecosystem, with many educational tools

The Davis In-Order (DINO) CPU: A Teaching-focused RISC-V **CPU Design** Authors: 🔊 🔤 WebRISC-V: a Web-Based Education-Oriented RISC-V Pipeline **Simulation Environment** WCAE'19: Proceed Authors: BRISC-V Platform: A Practical Teaching Approach for 1-8 • https://doi **Computer Architecture** WCAE'19: Proceed 1–6 • https://doi.or Authors: 🕿 Rashmi Agrawal, 🌊 Sahan Bandara, 麙 Alan Ehret, 🕿 Mihailo Isakov, 麙 Miguel Mark, 🧕 Michel A. Kinsy

WCAE'19: Proceedings of the Workshop on Computer Architecture Education • June 2019 • Article No.: 1 • Pages

• However, no RISC-V instruction converter yet!

RISC-V instructions

Authors Info & Claims

https://riscv.org/technical/specifications/

- <u>Volume 1</u>, Unprivileged Spec v. 20191213
- Volume 2, Privileged Spec v. 20211203

Types of RISC-V instructions

31	30	25	24	21	20	19	15	14	12 11	8	7	6	0	
funct7				rs2		rs	1	funct	3	ro	1	opo	code	R-type
	ir	nm[1]	:0]			rs	1	funct	3	rc	1	opc	ode] I-type
i	mm[11:5]			rs2		rs	1	funct	3	imm	[4:0]	opo	code	S-type
[imm[1	2] imm[1	0:5]		rs2		rs	1	funct	3 im	m[4:1]	imm[11]	opo	code	B-type
imm[31:12]									742	rc	l	opo	code	U-type
imm[2	0] ir	nm[1():1]	iı	nm[11] i	mm[1]	9:12]		rc	1	opo	code	J-type

 <u>Encoding</u> is from assembly form to binary form (e.g., addi a0, sp, 264 => 0x10810513) • <u>Decoding</u> is from binary form to assembly form (e.g., 0xFF740EE3 => beq s0, s7, -4)

rvcodec.js: An Educational Converter for **RISC-V Instructions**

SIGCSE 2023, March 15–18, 2023, Toronto, ON, Canada

Our tool: rvcodec.js

- Static single-page web application with no external dependencies
- User interface (UI) written in HTML + CSS + Javascript Handles user input and conversion parameters
- Renders converted output and colors matching fields
- Conversion engine written purely in Javascript • Contains encoding/decoding logic
- Builds Instruction object from user input
- Computes list of Fragments for coloring matching fields



Joël Porquet-Lupine 💁, Hikari N. Sakai, Abhi Sohal (and also Noah Krim and Dang Khoi Nguyen Ho)

export class Instruction // Instruction's properties and representations isa, fmt, len, asm, bin, hex, asmFrags, binFrags; // Decoding from binary to assembly

let decoder = new Decoder(this.bin, this.#config, this.#xlens);

export class Decoder #convertBinToAsm() // Detect OPCODE to run proper decoding logic this.#opcode = getBits(this.#bin, FIELDS.opcode.pos); switch (this.#opcode) case OPCODE.OP_IMM: this.#decodeOP_IMM(break #decodeOP_IMM() // Extract binary fields const fields = extractIFields(this.#bin); const imm = fields['imm'], rs1 = fields['rs1'], funct3 = fields['funct3'], rd = fields['rd']; // Convert fields to assembly const src = decReg(rs1), dest = decReg(rd); const immediate = decImm(imm); // Create fragments const f = { opcode: new Frag(TYPE_OP, this.#mne, this.#opcode, FIELDS.opcode.name), new Frag(TYPE_OP, this.#mne, funct3, FIELDS.funct3.name), new Frag(TYPE_RD, dest, rd, FIELDS.rd.name), new Frag(TYPE_RS1, src, rs1, FIELDS.rs1.name), new Frag(TYPE_IMM, immediate, imm, FIELDS.i_imm_11_0.name); this.binFrags.push(f['imm'], f['rs1'], f['funct3'], f['rd'], f['opcode']); this.asmFrags.push(f['opcode'], f['rd'], f['rs1'], f['imm']);

Features

- UI/UX

 - instruction
- Mnemonic completion
- Planned

 - hover

Project



• Conversion engine • Support for all of the base integer ISAs: RV{32,64,128}I + Zifencei + Zicsr • Support for all of the mainstream ISA extensions: M, A, F, D, Q, C

• Bright colors to visually map the relationships between an instruction's assembly tokens and binary fields Copy buttons for each representation of an

 Highlighting of matching assembly tokens and binary fields upon mouse hover • Tooltips explaining each binary field upon mouse

• Better responsive UI for small screens

• Tool available online https://luplab.gitlab.io/rvcodecjs



• Source code available under the GNU Affero GPL v3.0 We accept contributions! <u>https://gitlab.com/luplab/rvcodecjs/</u>

