master Go to file Code

lastmjs Merge pull request #37 fr... 8cc1478 · 4 years ago

elements	load hex into memory	7 years ago
redux	load hex into memory	7 years ago
.gitignore	add netlify headers	7 years ago
LICENSE	Initial commit	7 years ago
README.md	Update README.md	4 years ago
_headers	add netlify headers	7 years ago
index.html	begin to put in basic ...	7 years ago
package-lock.json	Update dependency ...	7 years ago
package.json	Update dependency ...	7 years ago
process.js	begin to put in basic ...	7 years ago
renovate.json	Add renovate.json	7 years ago
tsconfig.json	begin to put in basic ...	7 years ago
wasm-metal.d.ts	load hex into memory	7 years ago

README MIT license

ANNOUNCEMENT June 17, 2021: As you may have noticed, this project has not been worked on for a while. That would be a correct assessment. There has been no progress for a number of reasons, mostly due to the authors having other priorities. That doesn't mean the idea has died or was tried and failed. The project may be revived at some point in the future.

WASM Metal

A bare metal physical implementation of WebAssembly. That's right, a WebAssembly CPU. Inspired in part by this amazing talk.

GUI Microarchitecture Simulator

View the live demo.

```
git clone https://github.com/lastmjs/wasm-meta
cd wasm-metal
npm install
npm start
```

Go to http://localhost:5000 in your web browser.

RTL Microarchitecture Simulator

Not implemented yet.

Loading Microarchitecture Implementation to FPGA

Not implemented yet.

Roadmap

- [ ] Understand the WebAssembly ISA
- [ ] Implement the mircoarchitecture in HTML/CSS/JavaScript
  - This will be a GUI simulator of the microarchitecture
  - This simulator will become the specification for the microarchitecture
  - It will allow us to quickly experiment with hardware configurations
  - It will allow us to iterate and learn how the microarchitecture should work
- [ ] Implement the microarchitecture as an RTL design in an HDL
- [ ] Simulate the RTL design
- [ ] Test the implementation on an FPGA
- [ ] Design the ASIC

Why is this a good idea?

- Maybe it's not, but who cares
- The world is moving to WebAssembly
- Some of the biggest and potentially most world-changing projects are implementing their virtual machines as WebAssembly virtual machines (DFINITY, Ethereum)
- The bytecode is being designed as a compilation target for low-level languages first
- The bytecode is meant to execute at near-native speeds on a variety of underlying ISAs
- Java processors already offer potential benefits, even being compilation targets for a high-level language
- Cutting out the translation from WebAssembly to the underlying ISAs could provide efficiency benefits
- We could get rid of WebAssembly virtual machines entirely, and replace them with WebAssembly physical machines

About

A bare metal physical implementation of WebAssembly. That's right, a WebAssembly CPU.

Readme
MIT license
Activity
371 stars
12 watching
9 forks
Report repository

Releases

No releases published

Packages

No packages published

Contributors 3

lastmjs Jordan Last
renovate-bot Mend Renovate
electrovir electrovir

Languages

TypeScript 97.0%
HTML 2.3%
JavaScript 0.7%

- If all SIM cards are Java processors, imagine what WebAssembly processors could do. I'm betting that WebAssembly is a better bytecode than Java bytecode for what Java bytecode is doing with microcontrollers, so WebAssembly could take over all spaces that Java processors currently have

Prior art

- https://en.wikipedia.org/wiki/Jazelle
- https://stackoverflow.com/questions/1153076/does-android-castrate-the-arms-jazelle-technology
- https://en.wikipedia.org/wiki/Lisp_machine
- https://en.wikipedia.org/wiki/Java_processor
- https://en.wikipedia.org/wiki/High-level_language_computer_architecture