Boot Linux in a Web Browser on a Simulated RISC-V System

**Goals/System Specifications**
- Showcase RISC-V ISA
  - Interactive Linux session with minimal work on the part of the user (no installation of toolchains)
  - Promotion, Education
- Support RV64v43 IMA
  - (I) Want 64-bit Integer Support, but “Number” type in JS is 64-bit float – use Closure’s Long.js library
  - (M) Long.js lacks support for most required multiply/divide operations
  - (A) Single core, in-order, but required for boot
- Boot unmodified riscv-linux
  - Needs to be fairly performant (JavaScript + Long.js imposes a drastic performance penalty over the built-in number type)
  - Minimum 10 MiB of simulated memory for boot to ash shell with BusyBox Toolkit in initramfs
  - Interrupts/Timer support – difficult to predict how fast the simulated system is, but knowing is crucial for Linux boot, ESC key functionality
  - User Interaction through emulated terminal (inf-loop interpreting instructions freezes DOM updates)

**Boot Process**
- Download ~3MiB riscv-linux Kernel ELF from server
- JavaScript ELF Loader copies Kernel into Simulated Memory
- Handoff to Linux Kernel (instr. exec. loop)
- CPU Idle, no pending messages
- Wait for External Device Message
  - @ CPU Idle
- BCD, Special ESC Key Handling

**Execution Loop**
- Limitation of Webworker Message Passing: Communication between threads (eg. BCD, CPU) only works when receiver-thread is “waiting” – need to “pause” at cpu_idle in kernel, without breaking timers

**External Devices**
- Webworker 1: RISC-V CPU
- Webworker 2: Block Device
  - Block Device
  - Webworker 3: NIC
  - Webworker 4: Framebuffer

**Performance Optimizations**
- Initially, 10 seconds to boot minimal proxy-kernel and run basic C program
- Webworkers (new in HTML5)
  - “Standard” JS implementation is one-thread, cannot support simultaneous UI updates with a loop continuously executing instructions
  - Fix: use asynchronous setTimeout – but a minimum timeout of 4ms makes this infeasible (250 instructions)
  - Instead, UI (block character device) gets main thread, CPU + other devices each get a webworker thread
  - Inter-thread comm. with message passing
  - Reduced boot time from days to hours
- 32-bit Program Counter
  - 64-bit PC operations drastically increase seconds/cycle
  - Table of most significant 32 bits, indexed on most significant 12 bits of PC
  - Reduced boot time from hours to ~10 mins
- TLB
  - Translation process requires ~25 operations on 64-bit quantities per instruction fetch
  - “Infinite” sized TLB to avoid translation whenever possible
  - Reduced boot time from ~10 minutes to ~20 seconds
- Measured Specs on Chrome 35.0 on OSX
  - Execs 1.75 Million Instructions Per Second
  - 15 seconds to boot to ash shell prompt

**Future Work**
- Implement Additional Devices
  - Block Device: Programs like gcc won’t fit on initramfs
  - Framebuffer: boot a GUI
- Higher Performance
  - Ideally, JS 64-Bit Integer support
  - Support for 64-bit Integers in ASM.js
- Education
  - Web-based IDE for RISC-V Assembly, like MARS for MIPS