Plan 9 on RISC-V

Geoff Collyer

Plan 9 Foundation

ABSTRACT

We have ported Plan 9 to several early RISC-V 'Unix-capable' (RV64GCSU) implementations: the Microchip Polarfire Icicle, the *tinyemu* emulator, a pre-release Beagle V, and the SiFive HiFive Unmatched. The Beagle V and Unmatched ports are currently usable, but consume power when idle. This paper describes the porting process and makes recommendations.

1. Background

In July 2020, the Microchip™ Polarfire Icicle™ board[2] was due to be the first available RISC-V[6,3] system that looked capable of running a Unix-like operating system, including paging hardware, a gigabyte of RAM (which turns out to be actually 2GB in 2 banks), and gigabit Ethernet,[8] with multiple CPUs (cores, harts) capable of 64-bit and (in theory) 32-bit operation. It has one SiFive™ E51 RV64IMA core that lacks supervisor mode (a 'hobbled' hart) and starts the other four, which are SiFive U54 RV64GC cores at 600 MHz. The board contains no graphics hardware.

Richard Miller had a 32-bit RISC-V Plan 9[7] C compiler suite already, and was willing to create a 64-bit compiler suite. Without this, I would not have started this porting effort.

1.1. Timeline

I originally planned to port the Plan 9 *9k* kernel, which already ran on 64-bit `amd64` systems and could exploit a large address space, to RV32GC mode on the Icicle while Richard worked on the RV64 C compiler, then use the RV32GC port as a basis for an RV64GC port when the RV64 compiler was ready. The hardware was due to arrive in mid-September, which eventually slipped to mid-October.

In the meantime, we developed using the *tinyemu* emulator, which emulates both RV32 and RV64 architectures, though only a single processor. Richard ported it to Plan 9 and added a serial port and *virtio* Ethernet. I made small changes to it to improve debugging capabilities and it has proven helpful in finding bugs where the hardware's response has been to just sit there dumbly. Perhaps the SBI could accept requests on a UART to dump a hart's registers.

When the hardware arrived, we had a *9k* kernel running on *tinyemu*, but we then discovered some things that would require changes. *Tinyemu* starts our kernel in RISC-V 'machine' mode, in which paging is disabled, but all machine facilities may be configured. The lack of paging encourages starting RAM at `0x80000000` or higher, to match Unix kernel conventions. By early December, we had a 64-bit kernel running on one CPU of the Icicle board. By late December, all U54 CPUs were scheduling processes. A few C compiler fixes arrived through mid-January. After that, various mysterious misbehaviour disappeared. By early February, graceful reboot was working and by mid-February, *Sv48* paging was working. The system seems to be complete and solid enough to use as a CPU server, and should be relatively easy to adapt to future RV64G systems.

Since then, we have made minor fixes, code and performance improvements, and adapted to the pre-release Beagle V™ and its SBI and the Hifive Unmatched, including improving SoC (system-on-chip) configurability. In particular, self-checking code has been added to verify sanity in various conditions, and to attempt to tolerate the unexpected.

1.2. Position Statement

'Device trees', ACPI, (U)EFI and GUIDs are problems, not solutions, and we try to avoid them at all costs. (Why use meaningful names when you can use long, meaningless strings of hex digits?) The RISC-V Platform Specification subcommittee is just flat-out wrong to adopt every mistake ever made on the IBM PC or ARM. The RISC-V Platform Specification is disappointing; RISC-V presented an opportunity to rethink and replace this string of disasters.

> *How do you tell a bad standard? If it begins with ''I'', e.g, I2O, IPMI. If it ends with ''I'', e.g., ACPI, EFI, IPMI, etc. If it has the word ''intelligent'' in it, e.g., I2O, IPMI. Or, the best, if it has all three, e.g., IPMI.* – Ron Minnich

Also, life is too short to deal with SD and MMC cards, GPIOs, PHYs, I2C, SPI and other single-bit interfaces to guaranteed-model-specific hardware, and this crud shouldn't be necessary; hardware should be usable immediately after coming out of reset. If the BIOS or U-boot initialize devices sufficiently to use them, that's good enough.

1.3. Hardware and Firmware

A note on terminology: the *CLINT* is the per-CPU simple interrupt controller; the *PLIC* is the system-wide more-complex interrupt controller. The PLIC feeds into the CLINTs as the external interrupt signal. The *SBI*[4] is a sort of BIOS, but unlike a PC BIOS, it cannot be circumvented.

On the hardware, the boot ROM/flash starts OpenSBI which then starts *U-boot*, which starts our kernel in 'supervisor' mode, from which there is no escape, with additional undocumented restrictions:

- read-only access to the CLINT's timer registers;
- have to use SBI calls to set the CLINT timer (and maybe send and clear IPIs);

- SBI v0.2 HSM calls are not implemented in the provided Icicle OpenSBI;
- *u-boot* on the Icicle only starts all CPUs (*hart*s in RISC-V terminology) if one uses the `bootm` command with an FDT to run a *uImage* claiming to be a Linux kernel.

A result is that we can't switch the Icicle into RV32GC mode with the stock boot 'ROM', though it is possible in machine mode. So we abandoned the 32-bit port since it can't run on the available hardware, though it still ran in *tinyemu*, as the current wave of Unix-capable systems are all RV64GC, as will be any Unix-capable systems with more than 2GB of RAM.

There are conflicting accounts of the details of how RISC-V harts are started, particularly at what PC. *U-boot* on Icicle starts them all at once at the entry point in the *uImage* file, while the Beagle V's and Unmatched's *u-boot* starts only hart 1. There are other bits of ill-documented hardware:

- there's an L2 cache which adheres to RISC-V cache coherence principles, so can be largely ignored;
- the PLIC context ids apparently have consecutive values starting at 0: E51 hart 0 M (machine) mode, U54 hart 1 M, hart 1 S (supervisor) mode, hart 2 M, hart 2 S, hart 3 M, hart 3 S, hart 4 M, and hart 4 S. These should be predictable or discoverable without consulting a 'device tree'.

1.3.1. SiFive U740

On SiFive-U740-based systems (Beagle V and HiFive Unmatched™), use of the `WFI` instruction, instead of `PAUSE`, to save power when idling produces strange and varied behaviour: console serial output gets stuck, or time gradually stops advancing, or the system becomes very busy, possibly servicing interrupts. Without the Unmatched hardware reference manual, it's difficult to understand what is going wrong.

1.3.1.1. Beagle V

The now-cancelled Beagle V has 8 GB of RAM, and an L2 cache that is *not* coherent with DMA, thus requiring manual cache flushing, unlike the Icicle and Unmatched. (This was claimed to be a bug that would be fixed in production hardware, which will now never appear.) It also has a newer OpenSBI implementation that provides the HSM (hart state management) operations.

1.3.1.2. HiFive Unmatched

OpenSBI's *sbi_get_hart_status* appears to often report the wrong hart as the sole started hart.

1.4. RISC-V Peculiarities

Memory alignment requirements are stricter than most people are used to: natural alignment for scalars up to and including `vlong`†. Otherwise, we get

† On Plan 9, `vlong` is `long long`, which is always 64 bits.

alignment exceptions. The 64-bit compiler promotes most scalars to `long` when pushing them as function arguments, only `vlong`s, `double`s, pointers, and some `struct`s are wider. However, there can be gaps on the stack, e.g., when pushing an `int` then a pointer.

On the Icicle and Unmatched, all CPUS, memory caches and DMA accesses are coherent, which is a delight. The RISC-V specifications encourage this, but it is nevertheless unusual, surprising and noteworthy.

2. Plan 9 Changes

These are largely confined to the architecture-dependent source directories.

2.1. Removed Assumption of Memory at Address Zero

The original *9k* assumed that RAM started at physical address 0, and it took some trial-and-error to find and repair all such dependencies.

2.2. No Virtual Page Table

The technique of the ‘virtual page table’[5] (VPT), which injects the page table into itself as a top-level PTE, is used in the `386` and `amd64` ports, but appears to be inapplicable to RISC-V. So some page table accesses had to be made explicit, which is clearer anyway (the existing VPT code is obscure).

2.3. Variable Page Sizes and Page Table Levels

The system implements *Sv39*, *Sv48*, *Sv57*, and *Sv64* paging, where available. The supported hardware implements only *Sv39* in RV64, but *tinyemu* implements *Sv48* too. *Sv57* and *Sv64* are untested to date, but are straight-forward extensions from *Sv48*. (It should be possible in supervisor mode to discover the maximal paging scheme supported without constructing a suitable identity-map page table and without consulting a ‘device tree’.)

2.4. SoC Configuration

As usual, configuration for a new SoC requires checking addresses at the top of `mem.h`, but now the `conf` sections of kernel configuration files include descriptions, in C, of the SoC’s devices. The appendix contains an example of the 64-bit *tinyemu* configuration. See `tecpu` and `pfcpu` for complete examples.

2.5. Starting CPUs During Bootstrap

On x86 systems, a single CPU starts at bootstrap, and it then starts the others. RISC-V systems may start CPUs (*hart*s) at any time. The Icicle starts them all at once when U-boot’s `bootm` command starts the kernel, which is necessary because its SBI lacks the HSM (Hart State Management) commands that would otherwise be needed. The Beagle V and HiFive Unmatched both start a single CPU (or at least try to), which uses the SBI HSM calls to start the others. Plan 9’s start-up code now copes with those possibilities, and the situation of having just been restarted via `/dev/reboot`.

2.6. A C Idiom

In a C expression such as in the following, using Plan 9 types:

```
    uvlong uvl, va;
    uvl &= ~((1<<5) - 1);       /* zero low 5 bits */
    uvl = va & ~((1U<<12) - 1);/* get pure page number */
```

the result will probably not be what was intended. The ~ operator will have an `int` or `uint` operand, yielding a result of the same type, 32 bits wide. This result will be widened for the & or &= operator, but it may be zero-extended, thus ensuring that the result in `uvl` will have zeroes in its upper 32 bits. In particular, 64-bit physical addresses of RAM on RISC-V were being truncated. *6c* and now *jc* detect this inadvertent zero-extension in the `uint` case.

Ensuring that the operand (and thus result type) of ~ is `vlong` or `uvlong` avoids this problem. We have made this change throughout *9k*.

3. Performance

These are times to build the Plan 9 `rv` kernel from scratch mostly on RV64GC systems with 1Gb/s Ethernet using the same 10Gb/s Ethernet file server, except as noted. These were all effectively diskless, as is normal for Plan 9 systems. The commands executed were "`mk clean; mk; mk clean; time mk >/dev/null`" to load caches before measuring.

```
  1.07u  1.73s   2.04r mk # 4-core amd64 Xeon 3.8GHz, 10GbE
  5.66u  3.60s   9.35r mk # 4-core arm raspberry pi 4 1.5GHz
 10.91u  7.79s  11.46r mk # 4-core dual-issue hifive unmatched 1.2GHz
 14.99u  9.11s  50.91r mk # 2-core dual-issue pre-release beagle v
                          # 1GHz (diff. fs)
 23.49u 11.27s  19.64r mk # 4-core single-issue Icicle 600MHz
281.80u 90.86s 464.07r mk # 1-core tinyemu on Xeon 3.1GHz
```

See this earlier paper[1] for comparison with older Plan 9 systems of various architectures. The Icicle times are not spectacular, but its CPU cores are single-issue and not terribly fast.

4. Recommendations and Observations

Microchip's documentation seems to be unclear if it's intended for someone repackaging the hardware or for the ultimate end user. It often specifies that some value is programmable but doesn't provide the choice of value used in the Icicle. It would be helpful to have end user documentation.

The RISC-V architecture tries to leave some things unspecified to allow implementations some leeway, requiring that platform documentation provide the actual values implemented, but the platform makers don't always do so. Concern for RISC-V implementors should be balanced with concern for users; vagueness is rarely useful to system programmers. It would be more helpful to be able to query such values programmatically without consulting a 'device tree'.

All the timers provided require *a priori* knowledge of their frequencies. To let software determine the actual frequencies, it would be very helpful to have a real-time clock that ticks at a known, fixed rate (e.g., 100 times per second) or a register containing the (fixed) CLINT timer frequency. As it stands, the frequencies have to be supplied to software.

Detecting and reporting infinitely-recursive traps (perhaps in SBI) would be quite helpful during development. We have modified *tinyemu* to do this.

Requiring all RISC-V systems (or at least Unix-capable ones) to have an 8250-compatible console UART at a common, fixed physical address and a common frequency would help with porting. SiFive has its own non-8250-compatible UART.

Micro-USB connectors need to be braced very firmly; a slight tug on an attached cable should not yank the connector off the board. The current Unmatched board is quite flimsy. Its SD card slot isn't much better.

The Icicle's power cable is fragile and prone to interrupting power when flexed.

Suppliers need to implement both PXE booting and the `saveenv` command in their U-boot variants from the very start. These are important capabilities for kernel developers and must not be pushed off into the future. The Icicle at least has working PXE booting on one Ethernet, but no `saveenv` command, so unattended booting of non-default kernels won't work.

4.1. Assessment of RISC-V

In general, RISC-V seems to be a pleasant architecture with a few minor infelicities. (Implementing graceful reboot on the Icicle was a challenge.) SBI is another story.

The CSRR* instructions hard-code the CSR number; they would be easier to invoke from C if the CSR number were held in *rs2* instead, thus allowing use of less assembly language.

If the kernel's stack pointer contains an invalid address (e.g., a change in page tables makes it invalid), the trap to report the invalid address will trap endlessly due to an invalid stack pointer. SBI could perhaps note and report this.

It would be useful in a few cases to be able to determine the privilege mode, even if it's virtualized.

Machine mode seems dubious. Supervisor mode should be able to control the (possibly virtual) machine, and a mode without the possibility of paging is not helpful.

4.1.1. SBI

SBI seems largely unnecessary yet it insists on disabling some hardware features that a kernel could use directly and requiring use of SBI instead. I don't want or need another layer of software between the hardware and my kernel. The SBI specification is imprecise. For example, what are the units of the timer functions? Which timer do they set? What is that timer's frequency? Is the timer global or per-hart? Under what conditions can *sbi_send_ipi* (FID 0, EID 4) fail? It has been seen to fail with valid hart ids on OpenSBI. Which supervisor-mode facilities has it disabled?

There is some evidence of bugs in OpenSBI calls, e.g., *sbi_get_hart_status*.

5. Future Work

Bootstrapping is clumsy; a future upgrade to the Icicle's and Unmatched's *u-boot* should yield an automatic way to PXE boot at power-on. (Until then, *fshalt*(8) provides graceful reboots.)

There is Icicle and Unmatched hardware that we do not (yet) drive: an open PCI-E slot, an FPGA on the Icicle, and USB controller(s). Icicle documentation for USB is not obviously locatable. Icicle PCI-E requires newer HSS firmware.

6. Acknowledgements

Richard Miller developed the 32-bit and 64-bit RISC-V C compiler suites for Plan 9. He has been very helpful, fixing (minor) bugs, helping to find my obscure bugs, contributing an *sdio/mmc* driver, and extending the assemblers. The late Jim McKie created the 64-bit *9k* kernel and Charles Forsyth created the `amd64` compiler suite for the first architecture. We are building, of course, on years of work at Bell Labs creating and developing Plan 9.

References

1. Geoff Collyer, ''Recent Plan 9 Work at Bell Labs,'' *Fifth International Workshop on Plan 9*, Seattle, invited talk, `http://-www.collyer.net/who/geoff/ports.pdf`(October 2010).
2. *Icicle*, `https://www.microsemi.com/products/fpga-soc`.
3. *RISC-V home*, `http://riscv.org`.
4. *RISC-V Supervisor Binary Interface Specification*, `https://-github.com/riscv/riscv-sbi-doc/blob/master/-riscv-sbi.adoc`.
5. MIT, *Address translation and sharing using page tables*, `https://pdos.csail.mit.edu/6.828/2007/lec/-l5.html`.
6. David Patterson Andrew Waterman, *The RISC-V Reader,* Strawberry Canyon (7 November 2017). `http://riscvbook.com`
7. Rob Pike, Dave Presotto, Ken Thompson Howard Trickey, ''Plan 9 from Bell Labs,'' *Proc. of the Summer 1990 UKUUG Conf., London*, pp. 1-9 (July, 1990).
8. Xilinx, *Zynq 7000 SoC Technical Reference Manual (UG585)*, `https://www.xilinx.com/support/documentation/-user_guides/ug585-Zynq-7000-TRM.pdf`.

Appendix

```
/* 64-bit tinyemu configuration */
#include "riscv64l.h"
int cpuserver = 1;
int nosbi = 1;
uvlong cpuhz = 100*1000*1000;      /* emulated on xeon */
uvlong timebase = 10*1000*1000;    /* clint ticks per second */
Membank membanks[] = { PHYSMEM, GB, 0 }; /* (address, size) pairs */
char defnvram[] = "/boot/nvram";
uintptr uart0regs  =   PAUart0;
uintptr uartregs[] = { PAUart0 };
int nuart = nelem(uartregs);
vlong uartfreq = 384000;
uchar ether0mac[] = { 2, 0, 0, 0, 0, 1 };
/* the emulated plic doesn't seem to follow the spec; ignore it. */
Soc soc = {
        .clint   = (char *)PAClint,
        .uart    = (char *)PAUart0,
        .htif    = (char *)PAHtif,
};
Ioconf ioconfs[] = {      /* devices whose drivers vmap their regs */
        { "ether", 2*PGSZ, &soc.ether[0], 2, },
        0
};
Ioconf socconf[] = { /* devices without drivers that vmap their regs */
        { "clint", 64*KB, &soc.clint, },
        { "uart",  PGSZ, &soc.uart, 1, },
        { "htif",  PGSZ, &soc.htif, },
        0
};
void iophysaddrset(void) {}
```