Semihosting U-Boot

Look, ma, no serial!

Sean Anderson

SECO USA

FOSDEM 2023

How do you bootstrap a system?

- Fresh from factory
- Bricked

- Get code execution
- Write something to storage

Bootstrapping

- USB
  - DFU
  - UMS
  - SDP
  - Fastboot
- SD card
- JTAG
- Ethernet
  - TFTP
  - NFS
  - HTTP
  - Fastboot again
- Serial

What if you only have JTAG?

NXP QorIQ

- Communications processors
- Descended from 68k
- Lots of Ethernet, PCIe, and USB
- Hardware-accelerated networking

The Reset Configuration Word

- Started as basic initialization
- Pull-ups/downs on the boot bus
- Then they wanted pinmuxing
- And the RCW grew
- And moved to the boot device
- With a hard-coded RCW
- ...without runtime pinmuxing

The NXP solution

- Override the RCW using JTAG
- Partial reset and reboot
- These registers are undocumented
- Must use their JTAG probe
- With their IDE
- Not cheap!

A glowing review

Our manufactur[er] uses a single PC to perform the initial programming. On this PC, they have an evaluation copy of CodeWarrior. Every time that the evaluation copy expires... they erase the hard drive of the PC, install the OS again, and load another evaluation copy.

Enter semihosting

- Attach a debugger
- Execute a breakpoint instruction
- Opcode in `r0`
- Argument in `r1`
- Return code in `r0`
- Very similar to syscalls

So what do you get?

SYS_WRITEC

- `putchar(3)`
- One breakpoint and memory access per character
- Very slow (170 baud)

```
void puts(const char *s)
{
  while (*s)
    smh_trap(SYS_WRITEC, s++);
}
```

SYS_WRITE0

- puts(3)
- One breakpoint per string
- One memory access per character
- A bit faster (1600 baud)

```
void puts(const char *s)
{
  smh_trap(SYS_WRITE0, s);
}
```

SYS_WRITE

- `write(2)`
- One breakpoint per string
- Two memory accesses per string
- Reasonable speed (20000 baud)

```
void puts(const char *s)
{
  struct {
    long fd;
    void *buf;
    long len;
  } write = {
    .fd = stdout,
    .buf = s,
    .len = strlen(s),
  };
  smh_trap(SYS_WRITE, &write);
}
```

SYS_OPEN

- Any file on the system
- SYS_SEEK, SYS_READ, SYS_CLOSE, SYS_FLEN
- `load mmc 0 $loadaddr linux.img; bootm`
- Special file `:tt` is stdin/stdout

Initializing DRAM

- Load SPL, Initialize DRAM, Load U-Boot
- When do we load U-Boot?
- Dead reckoning
  - Variable runtime drives up timeout
- Reimplement DRAM init in TCL
  - No timing to worry about
  - Completely different process
- Semihosting makes this a non-issue

The rest

- Error handling (`SYS_ERRNO`, `SYS_ISERROR`)
- Time (`SYS_TIME`, `SYS_ELAPSED`)
- libc emulation (`SYS_GET_CMDLINE`, `SYS_HEAP`)
- Worrying (`SYS_SYSTEM`, `SYS_REMOVE`)

The two binary problem

- Breakpoints are invalid instructions...
- ...unless caught by the debugger
- Traditionally, two programs needed
- But with this one, simple trick...

Detecting a debugger

```
void do_sync(struct pt_regs *regs)
{
  if (ESR_ELx_EC(regs->esr) != ESR_ELx_EC_UNKNOWN)
    panic();
  if (*(u32 *)ALIGN_DOWN(regs->elr, 4) != SMH_A64_HLT)
    panic();
  disable_semihosting();
  regs->regs[0] = -1;
  regs->elr += 4;
}
```

How to semihost U-Boot

- CONFIG_SEMIHOSTING, CONFIG_SPL_SEMIHOSTING
- CONFIG_SEMIHOSTING_SERIAL, CONFIG_SERIAL_PUTS
- CONFIG_SEMIHOSTING_FALLBACK, CONFIG_SPL_SEMIHOSTING_FALLBACK
- RISC-V support from Kautuk Consul
- https://u-boot.readthedocs.io/en/latest/usage/semihosting.html
- ...and a debugger

OpenOCD

- Debug server for JTAG
- `halt`
- `arm semihosting enable`
- `resume`

- Uses the same terminal as messages
- Serial is cooked
- Single-threaded
- No sandboxing

Should you use semihosting?

- If you have to
- ...or if it's convenient
- Great if you're already booting with JTAG

Thanks

- Tom Verbeure
- Andre Przywara
- Tom Rini, Simon Glass, and others
- Marek Vasut
- SECO

Further reading

- https://tomverbeure.github.io/2021/12/30/Semihosting-on-RISCV.html
- https://github.com/riscv-software-src/riscv-semihosting
- https://developer.arm.com/documentation/dui0471/i/semihosting

Bonus line endings

- `"hello\n"`
- `'h', 'e', 'l', 'l', 'o', '\r', '\n'`
- `"hello\n", '\r'`
- `"hello", '\r\n'`