Digging for Documentation

A Tale of Reverse Engineering

Daniel Maslowski

Hello, I am Daniel :-)

Work and education

- IT security and computer science
- software engineer
- infrastructure and web
- apps, UIs, ecommerce

Open Source contributions

- hardware and firmware
- operating systems
- software distributions
- reverse engineering

Hello, I am Daniel :-)

Work and education

- IT security and computer science
- software engineer
- infrastructure and web
- apps, UIs, ecommerce

Open Source contributions

- hardware and firmware
- operating systems
- software distributions
- reverse engineering

I joined RISC-V International as an Individual Member.

Agenda

- Introduction
- Requirements
- Call to Action

Introduction

Documentation…?

Documentation…?

Documentation is needed everywhere.

[1] https://www.writethedocs.org/videos/eu/2017/the-four-kinds-of-documentation-and-why-you-need-to-understand-what-they-are-daniele-procida/

Documentation...?

Documentation is needed everywhere.

There are different ways to categorize kinds of documentation[1], e.g.:

- learning-oriented tutorials
- goal-oriented how-to guides
- understanding-oriented discussions
- information-oriented reference material

[1]https://www.writethedocs.org/videos/eu/2017/the-four-kinds-of-documentation-and-why-you-need-to-understand-what-they-are-daniele-procida/

Documentation…?

Documentation is needed everywhere.

There are different ways to categorize kinds of documentation[1], e.g.:

- learning-oriented tutorials
- goal-oriented how-to guides
- understanding-oriented discussions
- information-oriented reference material

We will look at ISAs as well as their implementations and applications.

[1] https://www.writethedocs.org/videos/eu/2017/the-four-kinds-of-documentation-and-why-you-need-to-understand-what-they-are-daniele-procida/

To Whom it Matters

Individuals

- RISC-V International Slack
- Telegram groups
- Matrix, e.g. Milk-V
- Reddit, /r/riscv
- Forums, e.g. StarFive
- OS distro communities

To Whom it Matters

Individuals

- RISC-V International Slack
- Telegram groups
- Matrix, e.g. Milk-V
- Reddit, /r/riscv
- Forums, e.g. StarFive
- OS distro communities

Organizations

- Academia, students & researchers
- Chip designers & vendors
- OEMs, e.g. Framework
- Industry groups, e.g. RISE
- OS distro vendors

Status Quo at RISC-V International

Status Quo at RISC-V International

RISC-V specifications are developed by groups and sources kept on GitHub.

Status Quo at RISC-V International

RISC-V specifications are developed by groups and sources kept on GitHub.

There is a template and a guide:

- https://github.com/riscv/docs-spec-template
- https://github.com/riscv/docs-dev-guide

Status Quo at RISC-V International

RISC-V specifications are developed by groups and sources kept on GitHub.

There is a template and a guide:

- https://github.com/riscv/docs-spec-template
- https://github.com/riscv/docs-dev-guide

PDFs are created from the spec releases:

- https://github.com/riscv/riscv-isa-manual/releases
- https://github.com/riscv-non-isa/riscv-sbi-doc/releases/tag/v2.0

Status Quo at RISC-V International

RISC-V specifications are developed by groups and sources kept on GitHub.

There is a template and a guide:

- https://github.com/riscv/docs-spec-template
- https://github.com/riscv/docs-dev-guide

PDFs are created from the spec releases:

- https://github.com/riscv/riscv-isa-manual/releases
- https://github.com/riscv-non-isa/riscv-sbi-doc/releases/tag/v2.0

This allows everyone to easily contribute, raise questions, add clarifications, and join the development.

Status Quo for Core and SoC Vendors

RISC-V cores are implemented based on the RISC-V specs.

Status Quo for Core and SoC Vendors

RISC-V cores are implemented based on the RISC-V specs.

Some implementations are open source, some are closed source, and some are derivatives of open source ones.

Status Quo for Core and SoC Vendors

RISC-V cores are implemented based on the RISC-V specs.

Some implementations are open source, some are closed source, and some are derivatives of open source ones.

Cores are used in SoC design, with peripherals added, often from third parties.

Status Quo for Core and SoC Vendors

RISC-V cores are implemented based on the RISC-V specs.

Some implementations are open source, some are closed source, and some are derivatives of open source ones.

Cores are used in SoC design, with peripherals added, often from third parties.

Documentation for cores and SoCs is, depending on the vendor,

- not publicly available
- available, but watermarked and restrictive
- published with delays or as drafts only
- sometimes in HTML, sometimes PDF form
- not possible to easily contribute to

Status Quo for OEMs and End Products

Status Quo for OEMs and End Products

More and more SBCs, tablets, laptops and other machines are being offered based on RISC-V cores.

- 2021: first SBCs
- 2024: multiple laptops

Status Quo for OEMs and End Products

More and more SBCs, tablets, laptops and other machines are being offered based on RISC-V cores.

- 2021: first SBCs
- 2024: multiple laptops

End customers have received unfinished products sold as “open source”.

Note: Sources were and are often not available, e.g. the SoC design / RTL.

Status Quo for OEMs and End Products

More and more SBCs, tablets, laptops and other machines are being offered based on RISC-V cores.

- 2021: first SBCs
- 2024: multiple laptops

End customers have received unfinished products sold as “open source”.

Note: Sources were and are often not available, e.g. the SoC design / RTL.

Depending on the vendors, they may upstream software such as U-Boot, Linux, and OpenSBI. In some cases, community members do the work.

Status Quo for OEMs and End Products

More and more SBCs, tablets, laptops and other machines are being offered based on RISC-V cores.

- 2021: first SBCs
- 2024: multiple laptops

End customers have received unfinished products sold as “open source”.

Note: Sources were and are often not available, e.g. the SoC design / RTL.

Depending on the vendors, they may upstream software such as U-Boot, Linux, and OpenSBI. In some cases, community members do the work.

Sometimes, short guides or manuals are available.

Schematics are mostly available, sometimes board designs.

Requirements

Layers of Documentation for Hardware

- ISA specs
  - see RISC-V International on GitHub :-)
- public core manuals
  - consider community contributions
  - highlight implementation specifics
- public SoC manuals
  - consider community contributions
  - third party components
- schematics and board designs
  - consider publishing sources
  - see also OSHWA certifications
- mask ROMs and their protocols
  - signing, loading and flashing firmware
- firmware
  - sources, licenses, howtos, behavior, integration
- OS distributions
  - see also their wikis

Full SoC and Core Manuals

- put them on GitHub, possibly as source
  - generate manuals from VHDL or similar
  - let community ask questions, file PRs/issues for clarification
  - use releases for artifacts, PDFs etc, link on vendor website
- including full memory map
- ID register(s), efuses
- all peripherals (yes, also DRAM)

[2] https://github.com/BPI-SINOVOIP/pi-u-boot/blob/2aa062b0567d3863d700021990de89ba6b616c8f/drivers/ddr/spacemit/k1x/ddr_init_asic.h#L76

Full SoC and Core Manuals

- put them on GitHub, possibly as source
  - generate manuals from VHDL or similar
  - let community ask questions, file PRs/issues for clarification
  - use releases for artifacts, PDFs etc, link on vendor website
- including full memory map
- ID register(s), efuses
- all peripherals (yes, also DRAM)

DRAM controllers

- see StarFive forum https://forum.rvspace.org/t/dram-training-and-calibration/3224
  - note: we do have open source code, but it is hard to understand due to lack of documentation
- DRAM blobs are blocking progress
  - example: SpacemiT K1 DRAM training blob in header file[2]
- consider purchasing from more open suppliers

[2] https://github.com/BPI-SINOVOIP/pi-u-boot/blob/2aa062b0567d3863d700021990de89ba6b616c8f/drivers/ddr/spacemit/k1x/ddr_init_asic.h#L76

How SoCs Boot

How SoCs Boot

Typical SoCs have early code in their mask ROM, sometimes also called BROM (boot ROM) or ZSBL (Zero Stage Boot Loader).

How SoCs Boot

Typical SoCs have early code in their mask ROM, sometimes also called BROM (boot ROM) or ZSBL (Zero Stage Boot Loader).

Boot ROMs may offer protocols for loading over UART or USB, which is great for development, e.g., Allwinner FEL, JH71x0 Xmodem.

How SoCs Boot

Typical SoCs have early code in their mask ROM, sometimes also called BROM (boot ROM) or ZSBL (Zero Stage Boot Loader).

Boot ROMs may offer protocols for loading over UART or USB, which is great for development, e.g., Allwinner FEL, JH71x0 Xmodem.

Depending on the platform design, multiple further stages are necessary.

How SoCs Boot

Typical SoCs have early code in their mask ROM, sometimes also called BROM (boot ROM) or ZSBL (Zero Stage Boot Loader).

Boot ROMs may offer protocols for loading over UART or USB, which is great for development, e.g., Allwinner FEL, JH71x0 Xmodem.

Depending on the platform design, multiple further stages are necessary.

Developers need documentation:

https://github.com/sipeed/LicheePi4A/issues/12

> *I want to know how* `brom` *load uboot image(emmc) to ram, because I'm try to upstream uboot. :) This is only vendor can know.*

Tracking Progress

Tracking Progress

Software upstream status

- https://linux-sunxi.org/Linux_mainlining_effort#Status_Matrix
- https://wiki.rvspace.org/en/project/JH7110_Upstream_plan
- TH1520: https://docs.google.com/spreadsheets/d/1WzTS8zJ9ZbmBz3CO-ApQPiAtg7gl_wJT2prRZQXDlqQ

Tracking Progress

Software upstream status

- https://linux-sunxi.org/Linux_mainlining_effort#Status_Matrix
- https://wiki.rvspace.org/en/project/JH7110_Upstream_plan
- TH1520: https://docs.google.com/spreadsheets/d/1WzTS8zJ9ZbmBz3CO-ApQPiAtg7gl_wJT2prRZQXDlqQ

Operating Systems

FreeBSD needs help; see Mitchell Horne at BSDCan Developer Summit
FreeBSD on RISC V - May 2024 FreeBSD Developer Summit
https://www.youtube.com/watch?v=O7zW7Z9U0ns

Call to Action

What we can do

What we can do

RISC-V International

- guides towards upstream OS and firmware
- call for examples/tutorials; tutorials.riscv.org / examples.riscv.org ?
- curate contributions
- set up repos like for specs

What we can do

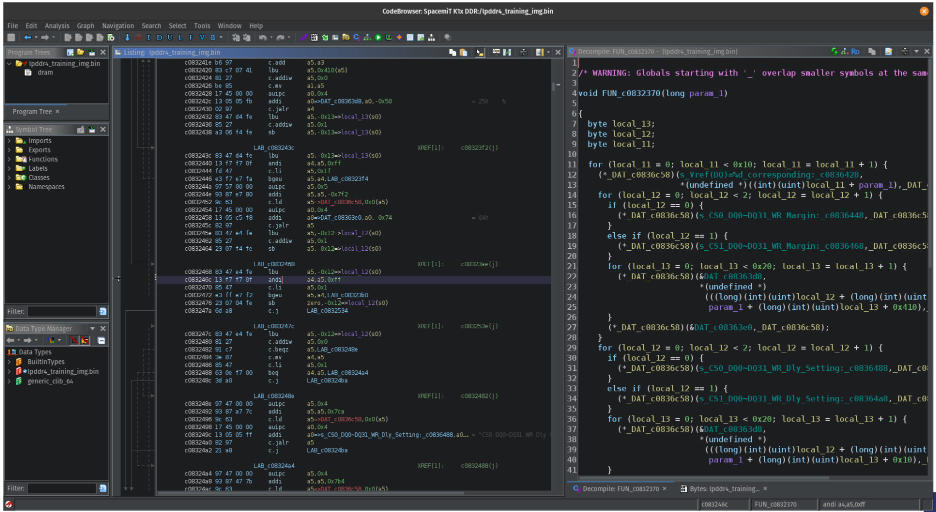
RISC-V International

- guides towards upstream OS and firmware
- call for examples/tutorials; tutorials.riscv.org / examples.riscv.org ?
- curate contributions
- set up repos like for specs

End Users

When you receive hardware, see how it works.
If you figure things out, share them with the community! :-)
Dump mask ROMs and other binaries to reverse them.

Ghidra?



```
c083241e b6 97        c.add     a5,a3
c0832420 83 c7 07 41  lbu       a5,0x410(a5)
c0832424 81 27        c.addiw   a5,0x0
c0832426 be 85        c.mv      a1,a5
c0832428 17 45 00 00  auipc     a0,0x4
c083242c 13 05 05 fb  addi      a0=>DAT_c08363d8,a0,-0x50            = 25h    %
c0832430 02 97        c.jalr    a4
c0832432 83 47 d4 fe  lbu       a5,-0x13=>local_13(s0)
c0832436 85 27        c.addiw   a5,0x1
c0832438 a3 06 f4 fe  sb        a5,-0x13=>local_13(s0)

                  LAB_c083243c                          XREF[1]:     c08323f2(j)
c083243c 83 47 d4 fe  lbu       a5,-0x13=>local_13(s0)
c0832440 13 f7 f7 0f  andi      a4,a5,0xff
c0832444 fd 47        c.li      a5,0x1f
c0832446 e3 f7 e7 fa  bgeu      a5,a4,LAB_c08323f4
c083244a 97 57 00 00  auipc     a5,0x5
c083244e 93 87 e7 80  addi      a5,a5,-0x7f2
c0832452 9c 63        c.ld      a5=>DAT_c0836c58,0x0(a5)
c0832454 17 45 00 00  auipc     a0,0x4
c0832458 13 05 c5 f8  addi      a0=>DAT_c08363e0,a0,-0x74            = 0Ah
c083245c 82 97        c.jalr    a5
c083245e 83 47 e4 fe  lbu       a5,-0x12=>local_12(s0)
c0832462 85 27        c.addiw   a5,0x1
c0832464 23 07 f4 fe  sb        a5,-0x12=>local_12(s0)

                  LAB_c0832468                          XREF[1]:     c08323ae(j)
c0832468 83 47 e4 fe  lbu       a5,-0x12=>local_12(s0)
c083246c 13 f7 f7 0f  andi      a4,a5,0xff
c0832470 85 47        c.li      a5,0x1
c0832472 e3 ff e7 f2  bgeu      a5,a4,LAB_c08323b0
c0832476 23 07 04 fe  sb        zero,-0x12=>local_12(s0)
c083247a 6d a8        c.j       LAB_c0832534

                  LAB_c083247c                          XREF[1]:     c083253e(j)
c083247c 83 47 e4 fe  lbu       a5,-0x12=>local_12(s0)
c0832480 81 27        c.addiw   a5,0x0
c0832482 91 c7        c.beqz    a5,LAB_c083248e
c0832484 3e 87        c.mv      a4,a5
c0832486 85 47        c.li      a5,0x1
c0832488 63 0e f7 00  beq       a4,a5,LAB_c08324a4
c083248c 3d a0        c.j       LAB_c08324ba

                  LAB_c083248e                          XREF[1]:     c0832482(j)
c083248e 97 47 00 00  auipc     a5,0x4
c0832492 93 87 a7 7c  addi      a5,a5,0x7ca
c0832496 9c 63        c.ld      a5=>DAT_c0836c58,0x0(a5)
c0832498 17 45 00 00  auipc     a0,0x4
c083249c 13 05 05 ff  addi      a0=>s_CS0_DQ0~DQ31_WR_Dly_Setting:_c0836488,a0... = "CS0 DQ0~DQ31 WR Dly
c08324a0 82 97        c.jalr    a5
c08324a2 21 a8        c.j       LAB_c08324ba

                  LAB_c08324a4                          XREF[1]:     c0832488(j)
c08324a4 97 47 00 00  auipc     a5,0x4
c08324a8 93 87 47 7b  addi      a5,a5,0x7b4
c08324ac 9c 63        c.ld      a5=>DAT_c0836c58,0x0(a5)
```

```c

/* WARNING: Globals starting with '_' overlap smaller symbols at the sam

void FUN_c0832370(long param_1)

{
  byte local_13;
  byte local_12;
  byte local_11;

  for (local_11 = 0; local_11 < 0x10; local_11 = local_11 + 1) {
    (*_DAT_c0836c58)(s_Vref(DQ)=%d_corresponding:_c0836428,
                     *(undefined *)((int)(uint)local_11 + param_1),_DAT_
    for (local_12 = 0; local_12 < 2; local_12 = local_12 + 1) {
      if (local_12 == 0) {
        (*_DAT_c0836c58)(s_CS0_DQ0~DQ31_WR_Margin:_c0836448,_DAT_c0836c5
      }
      else if (local_12 == 1) {
        (*_DAT_c0836c58)(s_CS1_DQ0~DQ31_WR_Margin:_c0836468,_DAT_c0836c5
      }
      for (local_13 = 0; local_13 < 0x20; local_13 = local_13 + 1) {
        (*_DAT_c0836c58)(&DAT_c08363d8,
                         *(undefined *)
                          (((long)(int)(uint)local_12 + (long)(int)(uint
                           param_1 + (long)(int)(uint)local_13 + 0x410),
      }
      (*_DAT_c0836c58)(&DAT_c08363e0,_DAT_c0836c58);
    }
    for (local_12 = 0; local_12 < 2; local_12 = local_12 + 1) {
      if (local_12 == 0) {
        (*_DAT_c0836c58)(s_CS0_DQ0~DQ31_WR_Dly_Setting:_c0836488,_DAT_c0
      }
      else if (local_12 == 1) {
        (*_DAT_c0836c58)(s_CS1_DQ0~DQ31_WR_Dly_Setting:_c08364a8,_DAT_c0
      }
      for (local_13 = 0; local_13 < 0x20; local_13 = local_13 + 1) {
        (*_DAT_c0836c58)(&DAT_c08363d8,
                         *(undefined *)
                          (((long)(int)(uint)local_12 + (long)(int)(uint
                           param_1 + (long)(int)(uint)local_13 + 0x10),_
      }
```

Vendor Specifics

Vendor Specifics

- StarFive
    - Provide source code for jh7110-recovery-20221205.bin
      https://github.com/starfive-tech/Tools/issues/2
    - Document XMODEM behavior with good/bad chips
      https://github.com/starfive-tech/Tools/issues/6
    - document OTP fuses and secure boot flow
      https://github.com/starfive-tech/Tools/issues/8
    - https://github.com/orangecms/vf2-header
    - https://github.com/orangecms/vf2-loader

Vendor Specifics

- StarFive
  - Provide source code for jh7110-recovery-20221205.bin https://github.com/starfive-tech/Tools/issues/2
  - Document XMODEM behavior with good/bad chips https://github.com/starfive-tech/Tools/issues/6
  - document OTP fuses and secure boot flow https://github.com/starfive-tech/Tools/issues/8
  - https://github.com/orangecms/vf2-header
  - https://github.com/orangecms/vf2-loader
- Bouffalo Lab https://github.com/orangecms/bouffalo-loader

Vendor Specifics

- StarFive
  - Provide source code for jh7110-recovery-20221205.bin https://github.com/starfive-tech/Tools/issues/2
  - Document XMODEM behavior with good/bad chips https://github.com/starfive-tech/Tools/issues/6
  - document OTP fuses and secure boot flow https://github.com/starfive-tech/Tools/issues/8
  - https://github.com/orangecms/vf2-header
  - https://github.com/orangecms/vf2-loader
- Bouffalo Lab https://github.com/orangecms/bouffalo-loader
- CVITEK/Sophgo https://github.com/orangecms/sg_boot

Vendor Specifics

- StarFive
  - Provide source code for jh7110-recovery-20221205.bin https://github.com/starfive-tech/Tools/issues/2
  - Document XMODEM behavior with good/bad chips https://github.com/starfive-tech/Tools/issues/6
  - document OTP fuses and secure boot flow https://github.com/starfive-tech/Tools/issues/8
  - https://github.com/orangecms/vf2-header
  - https://github.com/orangecms/vf2-loader
- Bouffalo Lab https://github.com/orangecms/bouffalo-loader
- CVITEK/Sophgo https://github.com/orangecms/sg_boot
- SpacemiT? TBD :-)

Thanks! :)

Follow Me

Daniel Maslowski

https://github.com/orangecms
https://twitter.com/orangecms
https://twitch.tv/cyrevolt
https://youtube.com/@cyrevolt

Related:

- Bootloaders in Limbo https://www.youtube.com/watch?v=-Ub8rCMrso0
- From Hardware Design to Rich OS with No Code https://www.youtube.com/watch?v=sxVIUeWkD6o

https://metaspora.org/digging-for-documentation.pdf