

---

# **Linux Openrisc Documentation**

**The kernel development community**

**Jul 14, 2020**



## **CONTENTS**



## OPENRISC LINUX

This is a port of Linux to the OpenRISC class of microprocessors; the initial target architecture, specifically, is the 32-bit OpenRISC 1000 family (or1k).

For information about OpenRISC processors and ongoing development:

website	<a href="http://openrisc.io">http://openrisc.io</a>
email	<a href="mailto:openrisc@lists.librecores.org">openrisc@lists.librecores.org</a>

---

### 1.1 Build instructions for OpenRISC toolchain and Linux

In order to build and run Linux for OpenRISC, you'll need at least a basic toolchain and, perhaps, the architectural simulator. Steps to get these bits in place are outlined here.

#### 1) Toolchain

Toolchain binaries can be obtained from [openrisc.io](http://openrisc.io) or our github releases page. Instructions for building the different toolchains can be found on [openrisc.io](http://openrisc.io) or Stafford's toolchain build and release scripts.

binaries	<a href="https://github.com/openrisc/or1k-gcc/releases">https://github.com/openrisc/or1k-gcc/releases</a>
toolchains	<a href="https://openrisc.io/software">https://openrisc.io/software</a>
building	<a href="https://github.com/stffrdhrn/or1k-toolchain-build">https://github.com/stffrdhrn/or1k-toolchain-build</a>

#### 2) Building

Build the Linux kernel as usual:

```
make ARCH=openrisc CROSS_COMPILE="or1k-linux-" defconfig
make ARCH=openrisc CROSS_COMPILE="or1k-linux-"
```

#### 3) Running on FPGA (optional)

The OpenRISC community typically uses FuseSoC to manage building and programming an SoC into an FPGA. The below is an example of programming a De0 Nano development board with the OpenRISC SoC. During the build FPGA RTL is code downloaded from the FuseSoC IP cores repository and built using the FPGA vendor tools. Binaries are loaded onto the board with `openocd`.

```
git clone https://github.com/olofk/fusesoc
cd fusesoc
sudo pip install -e .

fusesoc init
fusesoc build de0_nano
fusesoc pgm de0_nano

openocd -f interface/altera-usb-blaster.cfg \
        -f board/or1k_generic.cfg

telnet localhost 4444
> init
> halt; load_image vmlinux ; reset
```

#### 4) Running on a Simulator (optional)

QEMU is a processor emulator which we recommend for simulating the OpenRISC platform. Please follow the OpenRISC instructions on the QEMU website to get Linux running on QEMU. You can build QEMU yourself, but your Linux distribution likely provides binary packages to support OpenRISC.

qemu open- risc	<a href="https://wiki.qemu.org/Documentation/Platforms/OpenRISC">https://wiki.qemu.org/Documentation/Platforms/ OpenRISC</a>
--------------------	--

---

## 1.2 Terminology

In the code, the following particles are used on symbols to limit the scope to more or less specific processor implementations:

openrisc:	the OpenRISC class of processors
or1k:	the OpenRISC 1000 family of processors
or1200:	the OpenRISC 1200 processor

---

## 1.3 History

**18-11-2003 Matjaz Breskvar (phoenix@bsemi.com)** initial port of linux to OpenRISC/or32 architecture. all the core stuff is implemented and seams usable.

**08-12-2003 Matjaz Breskvar (phoenix@bsemi.com)** complete change of TLB miss handling. rewrite of exceptions handling. fully functional sash-3.6 in default initrd. a much improved version with changes all around.

**10-04-2004 Matjaz Breskvar (phoenix@bsemi.com)** alot of bugfixes all over. ethernet support, functional http and telnet servers. running many standard linux apps.

**26-06-2004 Matjaz Breskvar (phoenix@bsemi.com)** port to 2.6.x

**30-11-2004 Matjaz Breskvar (phoenix@bsemi.com)** lots of bugfixes and enhancements. added opencores framebuffer driver.

**09-10-2010 Jonas Bonn (jonas@southpole.se)** major rewrite to bring up to par with upstream Linux 2.6.36





**TODO**

The OpenRISC Linux port is fully functional and has been tracking upstream since 2.6.35. There are, however, remaining items to be completed within the coming months. Here' s a list of known-to-be-less-than-stellar items that are due for investigation shortly, i.e. our TODO list:

- Implement the rest of the DMA API...`dma_map_sg`, etc.
- Finish the renaming cleanup...there are references to `or32` in the code which was an older name for the architecture. The name we' ve settled on is `or1k` and this change is slowly trickling through the stack. For the time being, `or32` is equivalent to `or1k`.