Linux Openrisc Documentation

The kernel development community

Jan 15, 2023

CONTENTS

CHAPTER

ONE

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	https://openrisc.io
email	openrisc@lists.librecores.org

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 or our github releases page. Instructions for building the different toolchains can be found on openrisc.io or Stafford's toolchain build and release scripts.

binaries	https://github.com/openrisc/or1k-gcc/releases
toolchains	https://openrisc.io/software
building	https://github.com/stffrdhrn/or1k-toolchain-build

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 openrisc	https://wiki.qemu.org/Documentation/Platforms/OpenRISC

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 enhancments. 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

CHAPTER

TWO

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.

CHAPTER

THREE

FEATURE STATUS ON OPENRISC ARCHITECTURE

Subsystem	Feature	Kconfig	Status	Desc
core	cBPF-JIT	HAVE_CBPF_JIT	TODO	arch
core	eBPF-JIT	HAVE_EBPF_JIT	TODO	arch
core	generic-idle-thread	GENERIC_SMP_IDLE_THREAD	ok	arch
core	jump-labels	HAVE_ARCH_JUMP_LABEL	TODO	arch
core	thread-info-in-task	THREAD_INFO_IN_TASK	TODO	arch
core	tracehook	HAVE_ARCH_TRACEHOOK	ok	arch
debug	debug-vm-pgtable	ARCH_HAS_DEBUG_VM_PGTABLE	TODO	arch
debug	gcov-profile-all	ARCH_HAS_GCOV_PROFILE_ALL	TODO	arch
debug	KASAN	HAVE_ARCH_KASAN	TODO	arch
debug	kcov	ARCH_HAS_KCOV	TODO	arch
debug	kgdb	HAVE_ARCH_KGDB	TODO	arch
debug	kmemleak	HAVE_DEBUG_KMEMLEAK	TODO	arch
debug	kprobes	HAVE_KPROBES	TODO	arch
debug	kprobes-on-ftrace	HAVE_KPROBES_ON_FTRACE	TODO	arch
debug	kretprobes	HAVE_KRETPROBES	TODO	arch
debug	optprobes	HAVE_OPTPROBES	TODO	arch
debug	stackprotector	HAVE_STACKPROTECTOR	TODO	arch
debug	uprobes	ARCH_SUPPORTS_UPROBES	TODO	arch
debug	user-ret-profiler	HAVE_USER_RETURN_NOTIFIER	TODO	arch
io	dma-contiguous	HAVE_DMA_CONTIGUOUS	TODO	arch
locking	cmpxchg-local	HAVE_CMPXCHG_LOCAL	TODO	arch
locking	lockdep	LOCKDEP_SUPPORT	ok	arch
locking	queued-rwlocks	ARCH_USE_QUEUED_RWLOCKS	ok	arch
locking	queued-spinlocks	ARCH_USE_QUEUED_SPINLOCKS	ok	arch
perf	kprobes-event	HAVE_REGS_AND_STACK_ACCESS_API	TODO	arch
perf	perf-regs	HAVE_PERF_REGS	TODO	arch
perf	perf-stackdump	HAVE_PERF_USER_STACK_DUMP	TODO	arch
sched	membarrier-sync-core	ARCH_HAS_MEMBARRIER_SYNC_CORE	TODO	arch
sched	numa-balancing	ARCH_SUPPORTS_NUMA_BALANCING	—	arch
seccomp	seccomp-filter	HAVE_ARCH_SECCOMP_FILTER	TODO	arch
time	arch-tick-broadcast	ARCH_HAS_TICK_BROADCAST	TODO	arch
time	clockevents	!LEGACY_TIMER_TICK	ok	arch
time	context-tracking	HAVE_CONTEXT_TRACKING	TODO	arch
time	irq-time-acct	HAVE_IRQ_TIME_ACCOUNTING	TODO	arch
time	virt-cpuacct	HAVE_VIRT_CPU_ACCOUNTING	TODO	arch

Table 1 – continued from p

Subsystem	Feature	Kconfig	Status	Desc
vm	batch-unmap-tlb-flush	ARCH_WANT_BATCHED_UNMAP_TLB_FLUSH	—	arch
vm	ELF-ASLR	ARCH_HAS_ELF_RANDOMIZE	TODO	arch
vm	huge-vmap	HAVE_ARCH_HUGE_VMAP	TODO	arch
vm	ioremap_prot	HAVE_IOREMAP_PROT	TODO	arch
vm	PG_uncached	ARCH_USES_PG_UNCACHED	TODO	arch
vm	pte_special	ARCH_HAS_PTE_SPECIAL	TODO	arch
vm	THP	HAVE_ARCH_TRANSPARENT_HUGEPAGE	—	arch