How to port a new arch(nds32) to Linux mainline

Greentime Hu (胡英漢)

greentime@kernel.org

green.hu@gmail.com

greentime@andestech.com
Outline

• **Introductions**
  – About me
  – What is nds32(Andes)
  – Stories of nds32 Linux

• **Porting Linux to a new processor**
  – Prerequisite to port an arch to Linux mainline
  – What should you port for your arch

• **How to upstream your patchset**
  – Development cycle
  – Ready to mainline
  – Send the pull request to Linus

• **Reflections and Implications**
  – Thanks

• **Q and A**
Introductions
About me

- **Manager, Andes Technology Corporation (2008-2012, 2013-present)**
  - Linux kernel, RTOS, Arduino

- **Engineer, MediaTek (2012-2013)**
  - Linux kernel

- **National Cheng Kung University (2005-2007)**
  - Institute of Computer and Communication Engineering
  - 成功大學電腦與通信工程研究所

- **National Chengchi University (2001-2005)**
  - Department of Computer Science
  - 政治大學資訊科學系
What is nds32(Andes)

- Patented powerful 16/32-bit AndeStar™ RISC-like architecture
- 10 active AndesCore™: 2-8 stage pipeline, 1- and 2-issue
- Highly performance
  - Coremark: 5.41/MHz
  - DMIPS: 3.36/MHz
- Smaller code size
  - Code size of EEMBC automotive benchmark is 30% better than ARMv7m gcc
- Support of upstream mainline
  - Linux kernel, gcc, binutil, uboot, uclibc-ng, OpenOCD
- >140 licensees, >2.5B Andes-Embedded SoCs
- Taiwan Stock Exchange: 6533
Stories of nds32 Linux

• First version
  – 2.4
  – 2.6.x

• First Linux support since 2006

• Upstream binutil/gcc since 2014

• Upstream Linux kernel since 20171108

• Verifications
  – LTP, glibc testsuites, OpenPOSIX testsuites, busybox testsuites...
Why upstream your Linux

• **Pros**
  – Upgrade all API automatically
  – Get all the new features automatically
  – Save resources to sync the new version kernel
  – Review code strictly, higher reliability
  – Popularize the company
  – “If you are not using a stable/longterm kernel, your machine is insecure” – Greg KH

• **Cons**
  – Spend more time for community, reviewing patchset
  – Follow the rules

• **I think**
  – The sooner you do it, the better
Porting Linux to a new processor
Porting Linux to a new processor

- Porting Linux to a new processor architecture, part 1: The basics
  - https://lwn.net/Articles/654783/
- Porting Linux to a new processor architecture, part 2: The early code
  - https://lwn.net/Articles/656286/
- Porting Linux to a new processor architecture, part 3: To the finish line
  - https://lwn.net/Articles/657939/
Prerequisite to port an arch to Linux mainline

- Get to know your hardware
  - Virtual memory model
  - Format of the page table
  - Translation mechanism
  - VIVT/VIPT/PIPT
  - Cache/TLB operations
  - ASID/global page
  - Page attributes
Prerequisite to port an arch to Linux mainline

- Get to know your hardware
  - How to enable/disable interrupts
  - How to switch from privilege mode to user mode and vice-versa
  - How to get the cause of an exception
  - How to get the interrupt number
Prerequisite to port an arch to Linux mainline

• Get to know your hardware
  – What is ABI (Application Binary Interface)
    • Used for C code and assembly code
    • System call
    • Ftrace
    • Context switch
    • Caller/callee saved registers
Prerequisite to port an arch to Linux mainline

• Get to know the kernel
  – Low memory/high memory for 32bit CPU
  – Direct mapping/vmalloc regions/virtual memory layout
  – Kernel occupies the upper 1GB/1280MB(0xc0000000/0xb0000000)
  – kmap()/kmap_atomic() to gain temporary access to these high-memory pages

• A upstream toolchain
  – https://lkml.org/lkml/2018/2/26/77
  – “Removing architectures without upstream gcc support”
What should you port for your arch

- **arch/nds32**
  - **boot:** dts files
  - **configs:** a default configuration file
    - One kernel to run everywhere
  - **include:** header files for kernel or user space
  - **kernel:** functions for architecture and kernel
  - **lib:** optimized library
  - **mm:** functions for memory related features
  - **Kbuild**
    - Makefile
    - vmlinux.lds.S
      - #include <asm-generic/vmlinux.lds.h>
    - Kconfig/Kconfig.cpu
The header files

- `asm/` is part of the kernel interface and is used internally by the kernel source code.
- `uapi/asm/` is part of the user interface and is meant to be exported to user space.
- Use the generic one by Kbuild
  - `include/asm/Kbuild`
    - `generic-y += atomic.h`
    - `generic-y += barrier.h`
    - `...`
The header files

• Architecture specific
  – Cache(cacheflush.h, proc-fns.h, cache_info.h)
  – TLB management(tlb.h, tlbflush.h, mmu_context.h)
  – ELF format(elf.h)
  – IO operations(io.h, barrier.h)
  – Interrupt enable/disabling(irqflags.h, assembler.h)
  – Page table management(memory.h, page.h, pgalloc.h, pgtable.h, fixmap.h)
The header files

- **Architecture specific**
  - Context(mm_u_context.h, ptrace.h, processor.h, thread_info.h, mmu.h)
  - User space memory access(uaccess.h)
  - SYSCALL(unistd.h, syscalls.h, syscall.h)
  - VDSO(vdso_datapage.h, vdso.h, vdso_timer_info.h)
  - ATOMIC(futex.h)
  - MISC(nds32.h, swab.h, vdso.h, shmparam.h, dma-mapping.h, l2_cache.h, linkage.h, module.h, delay.h)
Boot sequence

• Boot from head.S
  – ENTRY(_stext)
  • before C code
    – Setup a temporary virtual memory
    – Setting system registers and clear bss sections
    – Set init_task(thread pointer) and stack pointer
    – b start_kernel
Boot sequence

- `start_kernel()`
  - `setup_arch()`
    - `early_init_devtree(__dtb_start)`
    - `setup_memory()` // Setup memblock
    - `paging_init()` // Create page table, allocate zero_page
    - `parse_early_param()` // To get boot_command_line
    - `unflatten_and_copy_device_tree()` // Copy and create tree of device_nodes
    - `early_trap_init()` // Copy vector table
  - `trap_init()` // Do nothing
  - `mm_init()`
    - `mem_init()` // Marks the free areas in the mem_map and tells us how much memory is free.
  - `init_IRQ()`
    - `irqchip_init()`
  - `time_init()`
    - `of_clk_init()`
  - ... // Init each sub system
  - `local_irq_enable()`
  - `rest_init()`
Create kernel threads

- Spawning kernel threads
  - start_kernel()
  - rest_init()
    - kernel_init: The first kernel thread
      » run_init_process(/init)
    - kthread:: To schedule a task to run
      » schedule() -> __schedule() ->
      context_switch() -> switch_to() ->
      __switch_to()
What shall we port for user space

• **System call**
  - To get the syscall number and jump to related syscall functions
  - Use `sys_call_table[___NR_syscalls]`
    • include/uapi/asm-generic/unistd.h

• **Signal**
  - Setup/restore signal context
  - Implement `sigreturn.S` syscall by VDSO

• **VDSO**
  - Support `sigreturn`, `gettimeofday`, `clock_getres`, `clock_gettime`
  - Create a share object for user to use
  - Also need to implement in glibc
How to upstream your patchset
Developing cycle

- Rebase to the latest kernel codes
- Refine your coding style
- Iterations
  - Prepare patchset
    - `git format-patch -o ./tmp/ --subject-prefix="PATCH v7" --cover-letter -n --thread=shallow -- cc="green.hu@gmail.com" 4959d43^..60f23e7`
  - Send patches
    - `git send-email --to greentime@andestech.com --to linux-kernel@vger.kernel.org --to arnd@arndb.de --to linux-arch@vger.kernel.org ./tmp`
  - Refine patches based on maintainers’ comments
Ready to be merged to linux-next

- Ask Stephen to pull your tree to linux-next
  - https://lkml.org/lkml/2018/2/21/81
- Apply a kernel.org account
  - https://korg.wiki.kernel.org/userdoc/accounts
  - https://www.kernel.org/category/faq.html
- Get your gpg key signed by 3 kernel developers
  - https://www.kernel.org/doc/ksmap/
Send your pull request

• Signed your tag of your tree

• Send the pull request to Linus
  – [GIT PULL] Andes(nds32) Port for Linux 4.17
  – Create Pull Requests
    • https://www.kernel.org/doc/html/latest/maintainer/pull-requests.html#create-branch
Reflections and Implications
Reflections and Implications

• A very interesting journey
• Win-win for customer, company, myself and Linux community
Thanks

- **My team member**
  - Vincent Ren-Wei Chen (陳人維)
- **My boss**
  - Wang, Tunghwa (王東華)
- **Reviewer**
  - Arnd Bergmann
References

- [PATCH 00/31] Andes(nds32) Linux Kernel Port
  - “overall this looks very nice, great work!”

- [PATCH v6 00/36] Andes(nds32) Linux Kernel Port
  - https://lkml.org/lkml/2018/1/18/118
  - “it's time to move this to the next step towards inclusion”
Q and A

• How many architectures are supported in the Linux kernel?
• What are the differences between upstreaming an architecture and a device driver?
• When is the best time to send a patch?
• What is the most difficult part of this process?
• When reviewers have different opinions?
• How long is the entire process?
LINUXCON
containercon
CLOUDOPEN

CHINA 中国

THINK OPEN
开放性思维