Porting Linux
to your favorite obscure
Arm SoC
Why?

- embedded Linux
- kernel 2.6.17
- I want a modern kernel
Preparation

● reference manuals, datasheets
● GPL software releases
  – may have to be requested first
● look at other chips, too
● write your own documentation
Serial ports

- very common in embedded devices
- root shell
- bootloader access
- can be hard to find
Bootloaders

- first code that runs
- basic hardware init
- loads the kernel and runs it
Bootloaders

- often have a serial port shell
- peek/poke
- storage access
- TFTP/xmodem/etc.
- run code

```
bootloader > h
W90P710 Command Shell v1.0 Rebuilt on Mar 23 2012 at 17:48:54
H       Display the available commands
B       Set Baud Rate
D       Display memory. D -? for help
E       Edit memory. E -? for help
G       Goto address
I       information
MX      Xmodem download
MT      TFTP download
FT      Program the flash by TFTP. FT -? for help
FX      Program the flash by Xmodem. FX -? for help
CP      Memory copy
LS      List the images in the flash
SET     Setting boot loader configuration. SET -? for help
CHK     Check the flash
RUN     Execute image
DEL     DEL the image or flash block
MSET    Fill memory
TERM    Change the terminal output port
BOOT    Reboot the system
CACHE   Cache setting
UNZIP   Unzip image
ATTRIB  Change the image attribution
MEMREAD Upload Read from memory
MEMWRITE Download write to memory sequence of values
```
Running your own code

- start simple
- use the serial port
- now you can run Linux :)

```
# scream on the debug uart.

mov r0, #0xb8000000
mov r1, #\'A\'

a:
  strb r1, [r0,#0x00]

b a
```
Monitor

- small program for experimentation
- peek/poke
- scripting
- complements the bootloader

Executing image 7 ...
Welcome to lolmon
> help
help - Show help output for one or all commands
echo - Echo a few words
rb - Read one or more bytes
rh - Read one or more half-words (16-bit)
rw - Read one or more words (32-bit)
wb - Write one or more bytes
wh - Write one or more half-words (16-bit)
ww - Write one or more words (32-bit)
cb - Copy one or more bytes
ch - Copy one or more half-words (16-bit)
cw - Copy one or more words (32-bit)
imb - Instruction memory barrier
call - Call a function by address
src - Source/run script at address
>  

Configuring Linux

- make ARCH=arm multi_v7_defconfig
- make ARCH=arm nconfig
- configure “System Type”
- trim unused drivers
Configuring Linux

- low-level debugging (DEBUG_LL)
- UART type and register width
- UART base address
- early printk
- built-in cmdline
  - earlyprintk debug
  - keep_bootcon
First boot

• load zImage into RAM, execute

Executing image 7 ...

Error: invalid dtb and unrecognized/unsupported machine ID
r1=0x00008000, r2=0x00000000
Available machine support:

<table>
<thead>
<tr>
<th>ID (hex)</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>ffffffff</td>
<td>Generic DT based system</td>
</tr>
</tbody>
</table>

Please check your kernel config and/or bootloader.
Devicetree

- hardware description
- “API” defined in DT bindings
- dts (source) → dtb (blob)
- enable ARM_APPENDED_DTB
- cat zImage foo.dtb > zImage-dtb
It boots... and then stops

Executing image 7 ... 
[ 0.000000] Booting Linux on physical CPU 0x0
[ 0.000000] Linux version 5.10.0-rc6-00003-g65c2ad7cf3e7 (jn@longitud
[ 0.000000] e) (arm-linux-gnueabi-gcc (Debian 10.2.0-9) 10.2.0, GNU ld (GNU Binutils
[ 0.000000] for Debian) 2.35.1) #14 PREEMPT Tue Dec 1 23:36:48 CET 2020
[ 0.000000] CPU: ARM926EJ-S [41069265] revision 5 (ARMv5TEJ), cr=0005
[ 0.000000] 317f
[ 0.000000] CPU: VIVT data cache, VIVT instruction cache
[ 0.000000] 0F: fdt: Machine model: Supermicro X9SCI-LN4F BMC
[ 0.000000] printk: bootconsole [earlycon0] enabled
[ 0.000000] Memory policy: Data cache writethrough
It boots... and then stops

- vendor kernel uses 100 MiB of RAM
- I used all 128 MiB
- hard to debug problems
- solution: only use 100 MiB
Timers

- Linux needs a timer interrupt to schedule tasks
- irqchip driver
- clockevent driver

rcu: RCU calculated value of scheduler-enlistment delay is 10 jiffies.
NR_IRQS: 16, nr_irqs: 16, preallocated irqs: 16
random: get_random_bytes called from start_kernel+0x2d4/0x4dc with crng_init=0
timer_probe: no matching timers found
Console: colour dummy device 80x30
printk: console [tty0] enabled
printk: bootconsole [earlycon0] disabled
Interrupt controller

- a.k.a. irqchip
- collects interrupts from peripherals
- CPU can handle them one by one
- with ARM Cortex-A*, usually ARM GIC
- up to ARM11, custom hardware
clockevent and clocksource

• Two kinds of timer-related drivers:
  – clockevent: Time-based interrupts
  – clocksource: Provides the current time

• The clk subsystem
  – manages oscillators, PLLs, dividers
  – required for some timer drivers
  – compatible = “fixed-clock”;
Boots to panic!

- That’s a success! :)
- It wants a root filesystem
Userspace

- init, shell, wget, you name it
- no storage drivers → initramfs (rootfs.cpio)
- buildroot
- console=ttyS0,115200
Ethernet

- more complex driver
- use wget+kexec to boot a new kernel

```bash
# wget 192.168.2.18:8000/arch/arm/boot/zImage 192.168.2.18:8000/arch/arm/boot/pci/
# nuvoton-wpcm450-supermicro-x9sci-ln4f.dtb
Connecting to 192.168.2.18:8000 (192.168.2.18:8000)
zImage 100% |*******************************************************************| 4341k  0:00:00 ETA
Connecting to 192.168.2.18:8000 (192.168.2.18:8000)
nuvoton-wpcm450-supe 100% |*******************************************************************| 2005  0:00:00 ETA
# kexec -l zImage --dtb nuvoton-wpcm450-supermicro-x9sci-ln4f.dtb
# kexec -e
[ 131.960000] kexec_core: Starting new kernel
[ 131.960000] Bye!
[ 0.000000] Booting Linux on physical CPU 0x0
[ 0.000000] Linux version 5.10.0-00014-ga321db916e9 (jn@longitude) (arm-linu
Many more drivers...
Things that can bite you

- Kconfig settings — better start from a defconfig
- using unavailable RAM
- kernel uses the wrong baud rate (9600)
- many other problems

Don’t give up, you can do it!
Upstreaming

- bringing your code to kernel.org
- via patches and mailing lists
- thorough review
- Greg KH – Write and submit your first kernel patch
Thank you.

GitHub: neuschaefoer/wpcm450
Freenode IRC: ##kernel

CC-BY 4.0