```
## Booting kernel from Legacy Image at a0000000 ...
   Image Name:   layerscape_kernel
   Created:      2017-09-28   9:35:51 UTC
   Image Type:   AArch64 RTEMS Kernel Image (uncompressed)
   Data Size:    42800 Bytes = 41.8 KiB
   Load Address: 80010000
   Entry Point:  80010000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
## Transferring control to RTEMS (at address 80010000) ...
[Kernel] Load module 'ARM-GICv3/4 Wrapper'...Success
[Kernel] Load module 'MMU Controller'...Success
[Kernel] Load module 'Exceptions'...Success
[Kernel] Load module 'Interrupt Handler'...Success
[Kernel] Load module 'Serial UART'...Success
[Kernel] Load module 'Hang'...Success
[Kernel] Load module 'Timer'...Success
[Kernel] Load module 'Syscall Controller'...Success
[Kernel] Load module 'IO Syscalls'...Success
[Kernel] Load module 'Memory Syscalls'...Success
[Kernel] Load module 'Thread Syscalls'...Success
[Kernel] Load module 'Process Scheduler'...Success
[Kernel] Load module 'Multicore-Support'...Success
[Kernel] Load module 'Performance Counter'...Success
[Kernel] Boot completed on all cores...
$:>
```

Layerscape Kernel

Dominik Walter

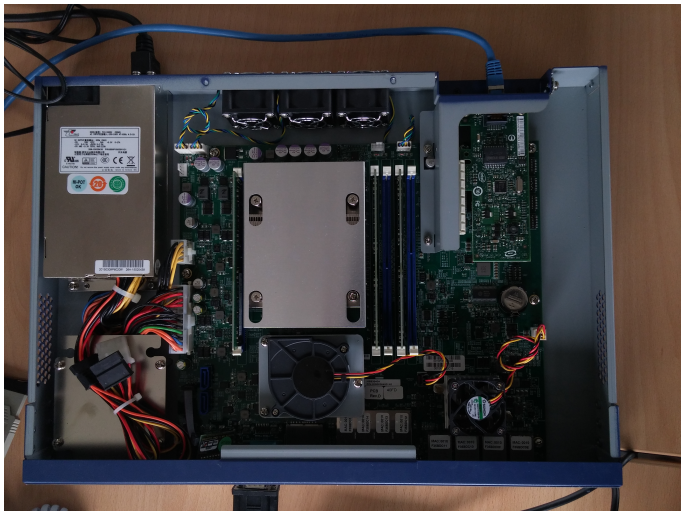
University of Augsburg

Systems and Networking

12.10.2017

General

- *freescale LS2085ARDB*
- 8x ARM Cortex-A57
- Data path acceleration architecture (DPAA2)
- 19 GiB RAM
- 128 MiB NOR Flash
- 2048 MiB NAND Flash



Project

Goal: Writing a small baremetal kernel for starting benchmark-applications.

- C++
- 3 Code-Projects
  - `layerscape_kernel`
  - `layerscape_shell`
  - `layerscape_helloworld`

Methode

How to write a kernel:

- Step 1: Search for Code-Examples
- Step 2: Read the Documentation
- Step 3: *"Try and Error"* until it works
- Step 4: Repeat

Installation

- `install.sh`
  - Download and install all required binaries.
- `configure.sh`
  - Set up the build configuration.

Required Packages/Binaries:

- *aarch64-elf/gcc-linaro-5.4.1-2017*
- *xinetd*
- *tftpd*
- *tftp*
- *u-boot-tools*
- *make*
- *cmake*
- *screen*

Build Process

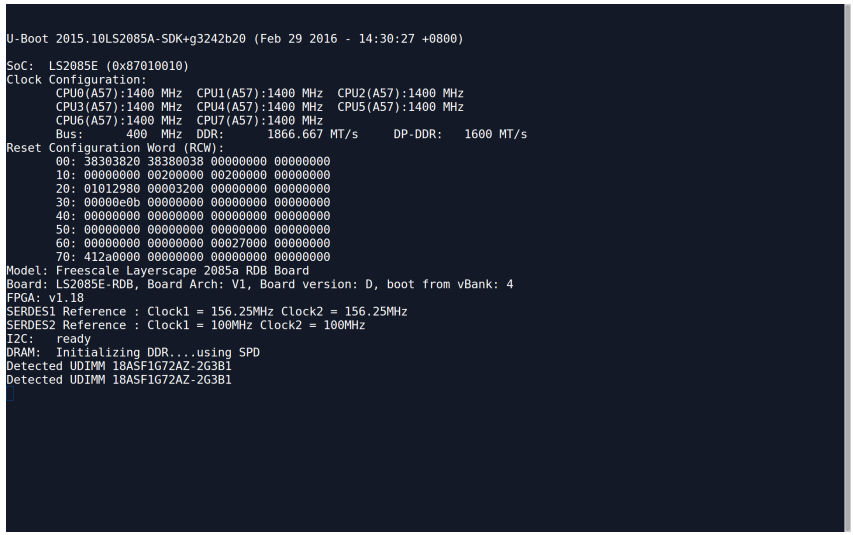
U-Boot

```
U-Boot 2015.10LS2085A-SDK+g3242b20 (Feb 29 2016 - 14:30:27 +0800)

SoC:  LS2085E (0x87010010)
Clock Configuration:
       CPU0(A57):1400 MHz  CPU1(A57):1400 MHz  CPU2(A57):1400 MHz
       CPU3(A57):1400 MHz  CPU4(A57):1400 MHz  CPU5(A57):1400 MHz
       CPU6(A57):1400 MHz  CPU7(A57):1400 MHz
       Bus:      400  MHz  DDR:      1866.667 MT/s     DP-DDR:   1600 MT/s
Reset Configuration Word (RCW):
       00: 38303820 38380038 00000000 00000000
       10: 00000000 00200000 00200000 00000000
       20: 01012980 00003200 00000000 00000000
       30: 00000e0b 00000000 00000000 00000000
       40: 00000000 00000000 00000000 00000000
       50: 00000000 00000000 00000000 00000000
       60: 00000000 00000000 00027000 00000000
       70: 412a0000 00000000 00000000 00000000
Model: Freescale Layerscape 2085a RDB Board
Board: LS2085E-RDB, Board Arch: V1, Board version: D, boot from vBank: 4
FPGA: v1.18
SERDES1 Reference : Clock1 = 156.25MHz Clock2 = 156.25MHz
SERDES2 Reference : Clock1 = 100MHz Clock2 = 100MHz
I2C:   ready
DRAM:  Initializing DDR....using SPD
Detected UDIMM 18ASF1G72AZ-2G3B1
Detected UDIMM 18ASF1G72AZ-2G3B1
```

Start U-Boot

- Plug the serial cable into the `UART`-Port 2
- Type `'screen /dev/ttyUSB0 115200'`
- Press the Power-Button to boot into U-Boot



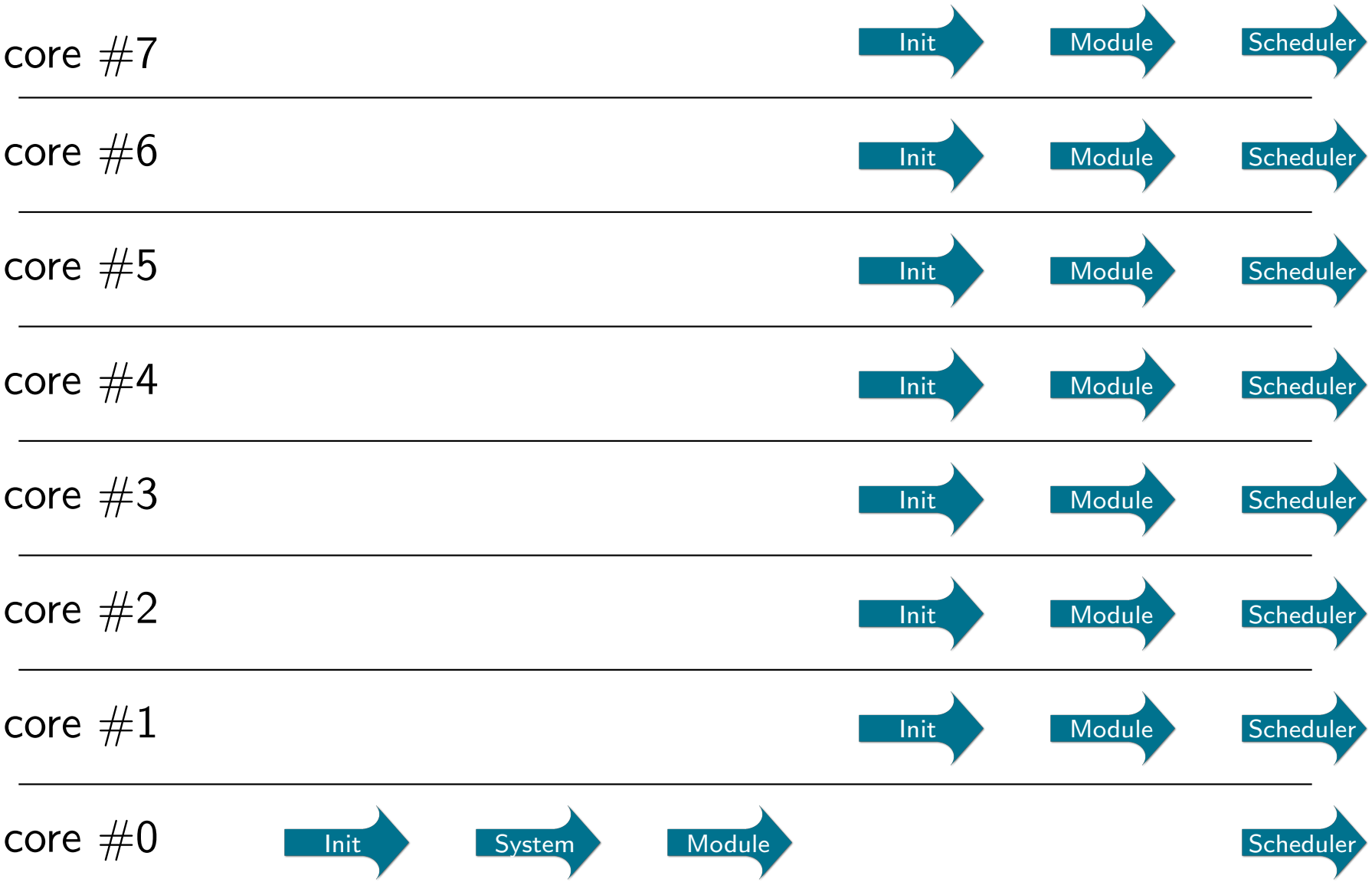
Deploy images

- dhcp
- setenv serverip 137.250.172.39
- tftpboot file.uimg
- cp.b $loadaddr 0xa0000000 $filesize
- bootm 0xa0000000

```
Using e1000#0 device
TFTP from server 137.250.16.21; our IP address is 137.250.172.64; sending through gateway 137.250.168.250
Filename '/pxelinux.0'.
Load address: 0x80100000
Loading: *
TFTP error: 'File not found' (1)
Not retrying...
Using e1000#0 device
TFTP from server 137.250.172.39; our IP address is 137.250.172.64
Filename 'layerscape_kernel.uimg'.
Load address: 0x80100000
Loading: #########
         4.1 MiB/s
done
Bytes transferred = 42864 (a770 hex)

. done
Erased 1 sectors
Copy to Flash... 9....8....7....6....5....4....3....2....1....done
Saving Environment to Flash...
Un-Protected 1 sectors
Erasing Flash...
. done
Erased 1 sectors
Writing to Flash... 9....8....7....69....8....7....6....5....4....3....2....1....done
Protected 1 sectors
=>
=>
=> tftpboot layerscape_shell.img; cp.b 0x80100000 0xb0000000 $filesize;
Using e1000#0 device
TFTP from server 137.250.172.39; our IP address is 137.250.172.64
Filename 'layerscape_shell.img'.
Load address: 0x80100000
Loading: #
         1.2 MiB/s
done
Bytes transferred = 2608 (a30 hex)
=>
```

Boot Process



Applications

Universität Augsburg University

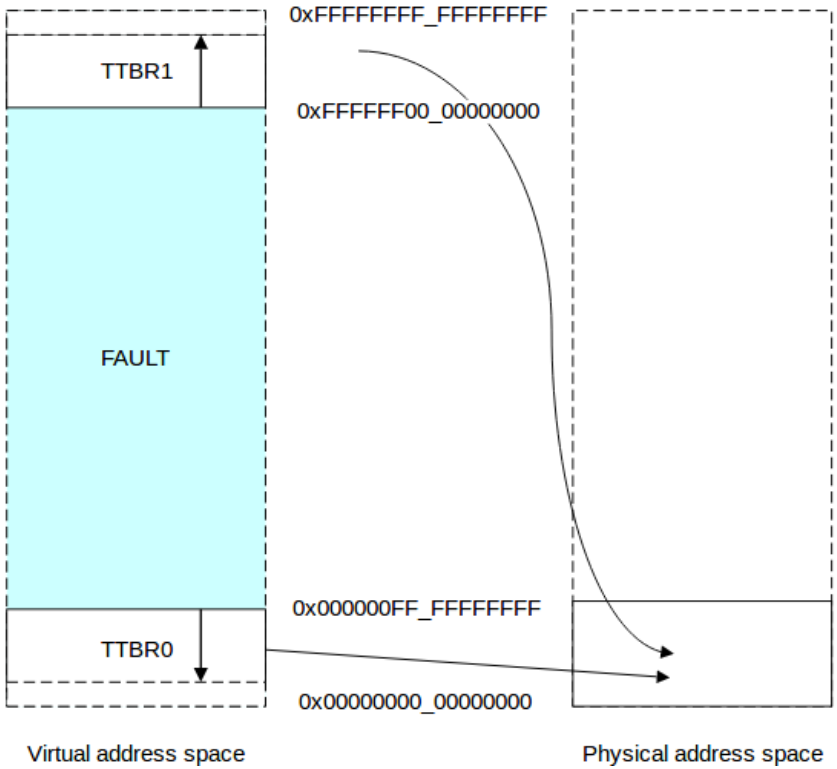
- Multiple Process-States (e.g. running, blocked)
- Primitive Signals (e.g. SIGTERM)
- Various Syscalls
- Unique Virtual Address Space
- Multithreading Support
- No POSIX!

Start Applications

- Memory region [0xB0000000, 0xBFFFFFFF] reserved for applications.
- .img-Format (no ELF-Loader)
- Load directly from Kernel with `create_process`.
  - Currently: Image and entrypoint at [0xB0000000] with a maximum size of 4KB.
- `layerscape_shell` provides a command-line-interface for loading applications.

Virtual Memory

- 40 Bit-Address
- 4KB Granularity
- Kernel: Leading 0s
- User: Leading 1s



Implemented Syscalls

Universität Augsburg University

- Input/Output
  - `read`
  - `write`
- Memory
  - `malloc`
  - `free`
- Process
  - `pid`
  - `thread_create`
  - `thread_join`
  - `process_create`
  - `EXIT`

Outlook

Further Ideas:

- Power Management
- Accelerator Unit
- Network-Stack
- Filesystem
- ELF-Loader
- Scheduling Algorithm
- Linux Kernelmodule Wrapper
- POSIX Compatibility