SPL on AArch64

Aaron van Geffen and Thom Wiggers

Institute for Computing
and Information Sciences
Radboud University

Why run SPL on ARM?

- Thom has lots of experience with ARM
- No more silly emulators in Java
- Seemed like fun

Real Hardware

Plaatje

- Quad-Core CPU
- 64-bit Cortex-A53
- ARMv8a
- 1.536 MHz
- 2GB RAM
- Low Power
- $ 45

AArch64

- 64-bit architecture mode of ARMv8a
- RISC architecture
- 31 general-purpose registers $r0, \ldots, r31$
  - Doesn't include the program counter or stack pointer (*sp*)
  - Notation: $x0$ is $r0$ in 64-bit mode, $w0$ is $r0$ in 32-bit mode
- Instructions have discrete inputs and outputs. They can also get one immediate values or receive their input with some rotation.
  - `add Rd, Rn, Rm`
  - `add Rd, Rn, #10`
  - `add Rd, Rn, Rm, LSL 10`
- Branches go through *Link Register*
- Advanced SIMD capabilities: 32 128-bit vector registers (not used in SPL)

AArch64 Calling Convention

We adopted ARM's C calling conventions[1] for function calls:

- Registers $r0, \ldots, r7$ are used for arguments
- Registers $r0, \ldots, r18$ are caller-saved
- Registers $r19, \ldots, r30$ are callee-saved
- Extra arguments are put on the stack
- The Link Register ($x30$) contains the return address
- Register $r0$ contains the function result after return

Allows easy use of libc library functions

[1] http://infocenter.arm.com/help/topic/com.arm.doc.ihi0055b/IHI0055B_aapcs64.pdf

Going from SSM to AArch64

SSM

- No registers used for operations
- Special operations for I/O
- Heap grows organically
- No separation of code and data segments

AArch64

- Mandatory registers for function calling
- System calls needed for I/O
- Heap allocated manually
- Specific code and data segments