ARMv8 mini-summit

Linaro Connect Copenhagen 2012

Andrew Thoelke, ARM Ltd

The Architecture for the Digital World®

Aims for today

- Inform
  - the status of open source software for ARMv8's 64-bit execution state
- Plan
  - the next quarter's work in Linaro (blueprints, requirements)
  - CI loop for 64-bit tools (gcc 4.7 etc.)
  - CI loop for 64-bit kernel
  - LAMP stack based on open embedded
- Coordinate
  - kernel activities for 32- and 64-bit architectures and platforms
  - 64-bit bring-up of distributions
- Enable
  - the wider development community

The Architecture for the Digital World®

ARMv8 Timeline

2007	ARM begins design of 64-bit Architecture
2009	ARM begins software development of 64-bit tools and kernel
Oct 2011	ARM announces ARMv8 at ARM TechCon 2011
Mar 2012	ARM & Linaro start planning of ARMv8 software rollout
Jun – Sep 2012	ARM & Linaro publish initial patches of tools and kernel
Sep 2012	Linaro bootstraps toolchain, kernel and OE stack from public source code
Oct 2012	... and publishes: http://www.linaro.org/engineering/armv8 ARM provides a free ARMv8 processor 'Foundation' model
2013	First silicon
2014	First products

The Architecture for the Digital World®

AArch64 upstream software status

	Target Version	Public/ Upstream	Notes
Linux kernel	3.7	Upstream	Maintainer: Catalin Marinas
Versatile Express ‘soc’	-	Published	In Catalin’s kernel.org git tree
gcc	4.8	Upstream	Co-maintainers: Richard Earnshaw and Marcus Shawcroft
binutils	2.23	Upstream	
newlib, libgloss	1.21	Upstream	
glibc	2.17	Published	Patches on public mailing lists
gdb	7.6	Published	Patches on public mailing lists
libffi	?	Published	Patches on public mailing lists
strace	?	Upstream	
UEFI	2.3.x	Q1’2013	In development

The Architecture for the Digital World®

Agenda

- Session 1: 09:00 – 09:55
  - arch/arm64 Linux Kernel
- Session 2: 10:00 – 10:45
  - Kernel cont'd
  - Booting and Firmware for AArch64
- Session 3: 11:00 – 11:55
  - AArch64 GNU Toolchain
  - AArch64 Developer Tools
- Session 4: 12:00 – 13:00
  - AArch64 Distributions and Community

The Architecture for the Digital World®
ARM®

The ARMv8 A64 Instruction Set

or

Where Have My Favourite ARM Instructions Gone?

Nigel Stephens, ARM Ltd

The Architecture for the Digital World®

A64 Development Process

- Work started in 2007
- Probably the best researched ARM ISA
- ISA and ABI prototyped in GCC and profiled on emulator
- Prototype CPU designed in parallel with ISA as it stabilised
- Further refined with help from lead architecture partners
  - announced and unannounced

The Architecture for the Digital World®

A64 Goals

- High-end "A-class" processors only
- Increase directly addressable physical and virtual memory for both kernel and user code
- Higher performance not a primary requirement
- *Static* code size not a primary requirement
- Focus on *dynamic* code size / instruction count in inner loops
  - Accept more instructions (larger code) in less executed areas
  - Reducing power consumption is also key for ARM

The Architecture for the Digital World®

Tweak or Clean Sheet?

- Large, flat virtual address space implies 64-bit registers and LP64 data model
- New register size & data model means a new ABI
  - Not going to be using legacy assembly code
- So a "clean sheet" ISA design would be possible
- But a 64-bit CPU must be an <u>excellent</u> 32-bit ARM CPU
  - Continue the ISA rationalisation begun by 32-bit Thumb
- Legacy break means opportunity for removing “cruft”

The Architecture for the Digital World®

Better use of Processor Resources

- ARMv7's execution modes with register banking means it has 31 general registers

R0	R1	R2	R3	R4	R5	R6	R7
R8	R9	R10	R11	R12	R13/SP	R14/LR	SP_hyp
LR_irq	SP_irq	LR_svc	SP_svc	LR_abt	SP_abt	LR_und	SP_und
R8_fiq	R9_fiq	R10_fiq	R11_fiq	R12_fiq	SP_fiq	LR_fiq	

- But only 14 (excluding SP & PC) allocatable by compiler
- Benchmarking shows significant benefit from exposing all

X0	X1	X2	X3	X4	X5	X6	X7
X8	X9	X10	X11	X12	X13	X14	X15
X16	X17	X18	X19	X20	X21	X22	X23
X24	X25	X26	X27	X28	X29	X30/LR	SP/ZERO

The Architecture for the Digital World®

Avoiding CPU Pinch Points

- The ARM ISA was designed for simple pipeline
  - In1999 ARM7 had a 3-stage pipeline @ 40 MHz
  - For comparison 1999 MIPS RM7000 ran @ 250 MHz
- A modern ARM CPU has complex ~15-stage pipe @ ~2 GHz
- Instruction set which works for leisurely 1999 pipeline is problematic for 2012 version, e.g.:
  - Predicated or conditional execution
  - Load/Store Multiple
  - Widespread access to PC (R15)
  - All register shifts on every arithmetic instruction
  - Arithmetic not updating all condition flags
  - Access to whole process state (CPSR and FPSCR)
  - Packed VFP / AdvSIMD registers

The Architecture for the Digital World®

But Look What We've Gained

- Optimised for modern OS platforms, languages, JITs & MP
  - Cleaner, more efficient ISA encoding
  - More useful immediate encodings
  - Larger PC-relative branch displacements
  - Vast inline PC-relative addressing
  - Unaligned addresses (almost) everywhere
  - 32 or 64-bit index register
  - IEEE754-2008 operations
  - Advanced SIMD usable for general-purpose floating point
  - Load-acquire and Store-release
  - Automatic "wakeup" events
  - User-level cache ops
  - Non-temporal load, store and prefetch

The Architecture for the Digital World®

Doesn’t it look a bit like MIPS?

- I couldn't possibly comment
- It has lost some idiosyncratic ARM features
- What remains is more like a "conventional" RISC ISA
  - So similar to MIPS, Alpha, PowerPC, HP-PA which all follow the same line of descent from Stanford RISC
- But clearly still an ARM instruction set
- I hope you enjoy programming with it!

The Architecture for the Digital World®

END

The Architecture for the Digital World®