VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Optimizing Embedded Software

A Look at the NEON SIMD unit in the ARM Cortex Family of Processors

Johan Dams

jd@puv.fi

VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Introduction

- Optimizing → making things faster, smaller
  - Why?
    - Lower power usage
    - Make better use of available capabilities

Sadly a lost/disappearing art...

- 80's home computer
  - Specialised chips for specific applications
    - Sound
    - Graphics
    - etc.

Embedded Systems Today

- More and more System On Chip
  - With integrated peripherals for specific purposes
    - Not unlike 80's computers – just smaller...
- Requires special skills to use
  - Compilers still not good enough for these extra 'cores'
  - Assembly still required (yes, this is 2009)

ARM / NEON

- ARM → general purpose CPU

- NEON → SIMD unit (Vector Processing Unit)
  - Part of Cortex architecture
  - Used for media and signal processing

i.MX515 Target Platform

- i.MX515 (Freescale)
  - Cortex A8 (800Mhz) + more (USB, Gfx, Audio)
  - Prototype (with screen) ~6W without power management
  - www.genesi-usa.com

Optimize What ?

- Linux Kernel?
  - Complex
  - Difficult to get involved
  - Can be quite hostile
  - Already quite optimized
- Individual Applications?
  - Often not possible (e.g., Android)

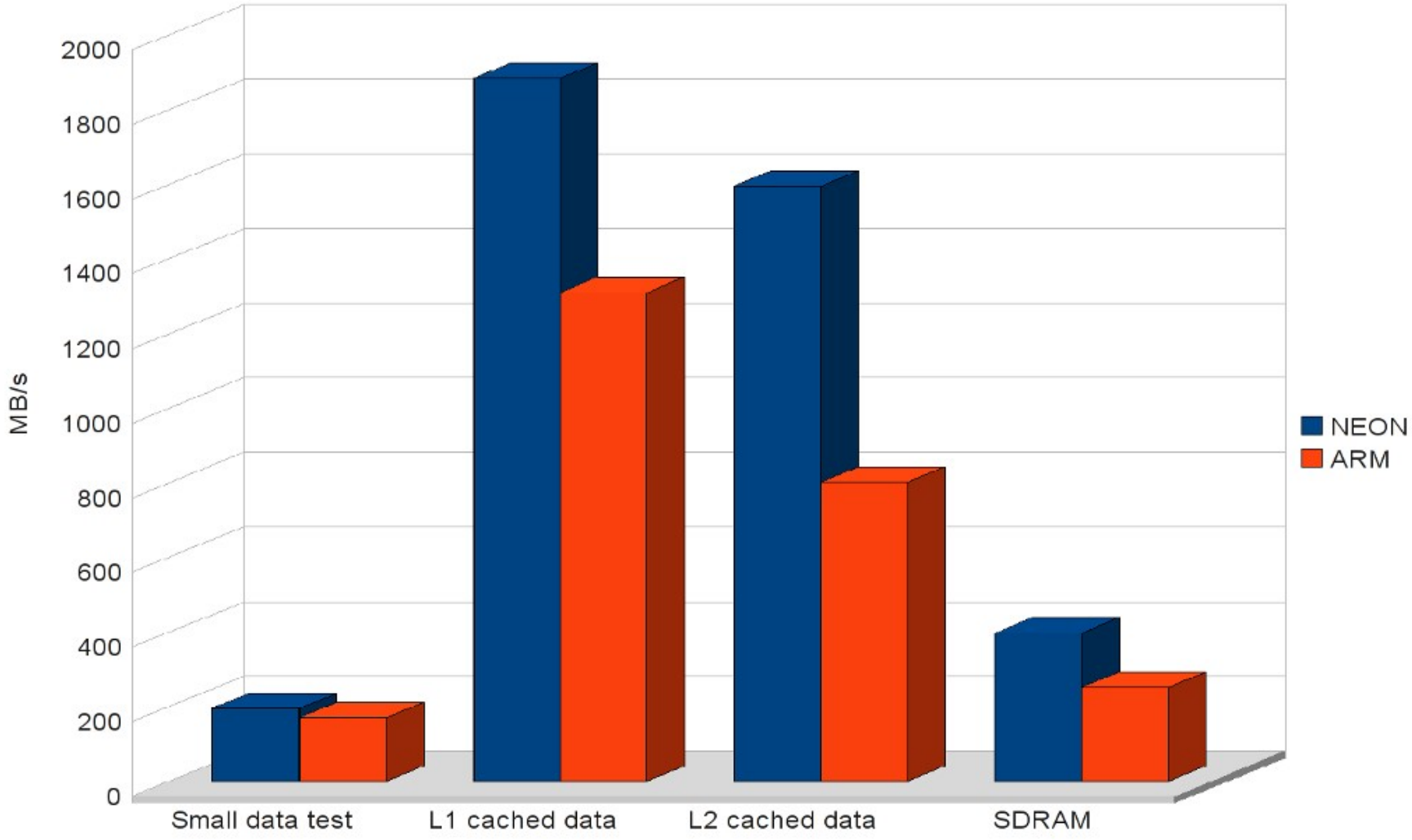
C Library

- Often overlooked!
- But:
  - Easy to get started
  - Benefits the entire system
  - Benefits existing applications without recompiling
- Successful Project as Reference
  - http://www.freevec.org

Starting Point?

- Memcpy()
  - Most used function
- Can extrapolate to other C library functions
  - Strcpy(), strlen(), memcmp(), memset(), etc.
- Also matrix operations
  - 4x4
  - VMAC

Memcpy() NEON Vs. Optimised ARM version



Standard C libraries

- Glibc, Eglibc, uClibc, etc...
  - Glibc = very generic
    - Very few optimizations
  - Others: build on Glibc code-base
    - Mostly optimize for size, not speed
- Many of the functions are SLOW
  - Much slower than the example in previous slide
- Reason: Glibc designed for servers/pc's in mind

Questions?

Thank You for Your Attention