RC2 - NEON-v8 programming

This course explains how to use ARMv8 NEON SIMD instructions to boost multimedia algorithms

Objectives

- This course has been designed for programmers wanting to run multimedia algorithms on NEON Single Instruction Multiple Data execute units on ARMv8 processors.
- Evolution of the NEON architecture between ARMv7 and ARMv8 is detailed.
- Each instruction family is detailed, first at assembly level, and then at C level using macros developed present in arm_neon.h file.
- Several tricky usage of processing instructions are provided.
- Vector and vector element load / store instructions are studied and guidelines for organizing data in memory are provided to minimize the number of memory accesses.
- The underlying cache operation as well as preload mechanisms (instruction and hardware prefetch) are detailed to explain how a processing can be pipelined .
- The course shows how DSP typical algorithms such as FIR and FFT can be vectorized and then optimized to be executed on NEON unit.
- Cryptographic operations are also detailed, with explanation of the supported algorithms.

Labs are compiled with GCC and run on a Linux Cortex-A53 board or a simulator
A more detailed course description is available on request at training@ac6-training.com

Prerequisites

- Knowledge of ARMv7 instruction sets.

Environnement du cours

- Cours théorique
  - Support de cours imprimé et au format PDF (en anglais).
  - Le formateur répond aux questions des stagiaires en direct pendant la formation et fournit une assistance technique et pédagogique.
- Activités pratiques
  - Les activités pratiques représentent de 40% à 50% de la durée du cours.
  - Elles permettent de valider ou compléter les connaissances acquises pendant le cours théorique.
  - Exemples de code, exercices et solutions
  - Un PC (Linux ou Windows) par binôme de stagiaires (si plus de 6 stagiaires) pour les activités pratiques avec, si approprié, une carte cible embarquée.
  - Le formateur accède aux PC des stagiaires pour l'assistance technique et pédagogique.
- Une machine virtuelle préconfigurée téléchargeable pour refaire les activités pratiques après le cours
- Au début de chaque demi-journée une période est réservée à une interaction avec les stagiaires pour s'assurer que le cours répond à leurs attentes et l'adapter si nécessaire

Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus.

Plan du cours

Day 1

Introduction to NEON

- Clarifying the resources shared by NEON and the scalar floating point engine
- Explaining the AArch32 and AArch64 differences
- NEON Register banks
- S, D and Q registers (AArch32)
- B, H, S, D and V registers (AArch64)
- Data types
- Vector vs scalar
- Related system registers
- Alignment issues
- Enabling NEON
- Differences between NEONv7 and NEONv8

NEON instruction syntax

- Instructions producing wider / narrower results
- Instructions modifiers
- Selecting the shape
- Selecting the operand / result type
- Syntax flexibility
- Declaring initialized vectors in C language
- Using unions with vectors and arrays of vectors to simplify the debug
- Casting vectors

Data transfer instructions

- Move
- Swap
- Table lookup
- Vector transpose
- Vector zip / unzip
- Data transfer between NEON and integer unit
- Practical lab: clarifying narrow and long instructions, building a vector from bytes selected from a pair of vectors
  Exercise : Example: managing audio samples
  Exercise : Using load with de-interleaving instructions to store all right lane samples into a vector and left lane samples into another vector
  Exercise : Clarifying narrow and long instructions, building a vector from bytes selected from a pair of vectors

Arithmetic Instructions

- Arithmetic instructions
- Add, modulo vs saturated arithmetic
- Halving / Doubling the result
- Rounding
- Subtract
- Multiply
- Multiply accumulate / Multiply subtract
- Absolute value
- Min / Max
  Exercise : Implementing a complex multiply accumulate with NEON

- Conversion instructions
- Converting Floating Point numbers into Fixed point numbers
- Converting Fixed point numbers into Floating point numbers
  Exercise : Converting fixed-point elements into single precision floating point values and adding the resulting elements
- Advanced arithmetic instructions
- Reciprocal estimate, reciprocal square root estimate, Newton-raphson algorithm
- Pairwise instructions

Day 2

Logic and Bitfield Instructions

- Element comparison
- Logic instructions
- Logical AND, Bit Clear, OR, XOR
- Operations with immediate values
- Bitfield instructions
- Count Leading zeros, ones, signs
- Bitwise insert instructions
- Conditional bitwise insert instructions, avoiding branches
- Shifts with possible rounding and saturation
- Bitfield reverse
  Exercise : Transposing a matrix, shifting a large bitmap using vector instructions

NEON Cryptography Extension

- The Cryptography extension
- Algorithms supported
- AES
- SHA1
- SHA256

Optimizing techniques

- Automatic vectorization
- Tuning loops for optimal results
- Avoid loop feedbacks
- Avoid loop-dependent conditionals
- Avoid early termination
- Padding loops
  Exercise : Experimenting with loop auto-vecorization
- Pointers and arrays
- indirect addressing
- pointer aliasing and restrict
  Exercise : Using restrict to eliminate dependencies
- Function calls and inlining
- promises
  Exercise : Making promises to help the compiler optimize
- Avoiding data dependencies

NEON coding examples

- FIR filter
- Converting the scalar algorithm into a vector algorithm
- Finding the NEON instructions to encode the vector algorithm
- Optimizing the code
- Using the performance monitor to tune the algorithm

- FFT (DFT)
- Converting the scalar algorithm into a vector algorithm, understanding how circle properties can be used to process 4 angles concurrently
- Finding the NEON instructions to encode the vector algorithm
- Optimizing the code
- Using the performance monitor to tune the algorithm