



## RC1 - NEON programming

**This course explains how to use 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.
- 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.
- This course has already been delivered to companies developing audio applications for mobile phones.
- THIS COURSE IS PROPOSED EITHER AS AN INSTRUCTOR-LED COURSE OR AS E-LEARNING.
- ACSYS has developed an optimized NEON based FFT coded in assembler language
  - performance for 1024 complex floating point single precision samples is:
    - - 106\_000 core clock cycles for Cortex-A9
    - - 90\_000 core clock cycles for Cortex-A8
  - for any information contact [guillaume.peron@ac6.fr](mailto:guillaume.peron@ac6.fr)

*Labs are run under RVDS*

*A more detailed course description is available on request at [info@ac6-training.com](mailto:info@ac6-training.com)*

### Prerequisites

- Knowledge of ARM V4T / V5TE instruction set.

## Plan

### DAY 1

#### **CORTEX-A8 AND CORTEX-A9(MP) ARCHITECTURE**

- Data path, studying how data are loaded from external memory and copied into level 1 and possibly level 2 caches
- Programmer's model
- Highlighting coherency issues when data are shared by several cores, purpose of the SCU implemented in Cortex-A9
- Cortex-A8 and Cortex-A9 instruction pipeline, branch predictors

#### **INTRODUCTION TO NEON/VFPv3**

- Clarifying the resources shared by NEON and VFP
- Register bank, Q registers, D registers
- Data types
- Vector vs scalar
- Related system registers
- Alignment issues
- Enabling NEON/VFP

#### **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

#### **LOAD / STORE INSTRUCTIONS**

- Addressing modes
- Vector load / store
- Vector load / store multiple
- Element and structure load / store instructions
  - Multiple single elements
  - Single element to 1 lane
  - Single elements to all lanes
- Optimizing the ordering of data in memory to take benefit of 2-, 3- and 4- element structures
  - Example: managing audio samples
- Processor acceleration mechanisms: store merging buffers
  - Practical lab: using load with de-interleaving instructions to store all right lane samples into a vector and left lane samples into another vector

### DAY 2

#### **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

## LOGICAL AND BITFIELD INSTRUCTIONS

- Logical AND, Bit Clear, OR, XOR
- Operations with immediate values
- Bitwise insert instructions, avoiding branches
- Count Leading zeros, ones, signs
- Normalizing floating point numbers when VFP is not implemented
- Scalar duplicate
- Extract
- Shift with possible rounding and saturation
- Bitfield revers
  - Practical lab: Transposing a matrix, shifting a large bitmap using vector instructions

## ARITHMETICAL INSTRUCTIONS

- Add, modulo vs saturated arithmetic
- Halving / Doubling the result
- Rounding
- Subtract
- Multiply
- Multiply accumulate / Multiply subtract
- Absolute value
- Min / Max
- Converting Floating Point numbers into Fixed point numbers
- Converting Fixed point numbers into Floating point numbers
- Reciprocal estimate, reciprocal square root estimate, Newton-raphson algorithm
- Pairwise instructions
- Element comparison
  - Practical lab: implementing a complex multiply accumulate with NEON
  - Practical lab: converting fixed-point elements into single precision floating point values and adding the resulting elements

## 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

## Renseignements pratiques

**Duration : 2 days**

**Cost : 1250 € HT**



SARL au capital de 15400€ - SIRET 449 597 103 00026 - RCS Nanterre - NAF 722C - Centre de Formation : 19, rue Pierre Curie - 92400 Courbevoie  
Siège social et administration : 21, rue Pierre Curie - 92400 Courbevoie - Tél. 01 41 16 80 10 - Fax. 01 41 16 07 78

Last site update: Tue 22 May 2012 10:50:29 AM CEST

<http://www.ac6-formation.com/>