



## AAA - ARM Cortex-A and R Architecture (v7/v8)

*This course explains the ARM Cortex-A and R global architecture.*

### Objectives

- Describing the ARM v7 and v8 architecture profiles A and R
- Describing the various ARM Cortex-A and R processor architecture
- Presenting the Hardware and Software implementation possibilities to learn how to create Cortex-A based applications
- Detailing ARMv8 Security (TrustZone) and Virtualization (Hypervisor) features
- This course provides all the prerequisites for the courses describing in details the various Cortex-A and Cortex-R cores and CPUs.

### Course environment

- Convenient course material with space for taking notes
- Demos targeting a Cortex-A9 based SoC

### Prerequisites

- Familiarity with embedded C concepts and programming
- Basic knowledge of embedded processors

### Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

## Course Outline

### First Day

#### Architecture

- Introduction to ARM and the Architecture
- The AArch32 Programmer's Model
- The AArch64 Programmer's Model
- Exceptions
- Memory Architecture
- Caches

#### Implementations

- Versions and Implementations
- ARMv4T
- ARMv5TE
- ARMv6
- ARMv7
- ARMv8
- SecurCore
- Architecture Extensions
- Pipelines

- Cycle Counting

## Second Day

### System features

- Multi-processing
- Cache maintenance
- Cache coherency hardware
- Interrupt distribution
- Power saving modes
- Memory system hierarchy
- Software storage and upload

### AArch64 Exception Model

- Four exception levels
- Exception Link Registers
- Register banking by exception level
- Nesting on the same exception level
- Exception type and exception origin
- Syndrome registers used to provide status information to the exception handler
- Exception return instruction
- AArch64 Exception vector tables

### Generic Interrupt Controller

- Generic Interrupt Controller CPU Interface Registers
- Interrupt Virtualization
- Interrupt Handling to support Nesting

## Third Day

### Multicore operation

- Single Processor / Multi-Task RTOS
- Multi-CPU Exclusive Resource Management
- Wait for Event / send Event
- Wait for Interrupt
- Multi-Processor / Multi-Task RTOS

### Software Development

- Embedded Software Development
- Libraries and Linkage
- Target platforms
- Memory ordering models
- Barriers and synchronization
- Cache policies
- Operating system support
- Booting

### Software Optimization

- Introduction
- Coding techniques
- Profiling

## Software debug

- Debug basics
- Debug hardware
- Invasive Debug
- Non-invasive Debug
- Standard Debug Techniques
- Timing
- Resources

## CoreSight Debug Components

- Self-Hosted Debug
- Debug State Instructions
- Linked comparisons for Breakpoint/Watchpoint exception generation
- Software Step exceptions
- Routing debug exceptions
- External debug, cross-triggering
- Embedded Trace Macrocell architecture

## Fourth Day

## ARMv8 Memory Management Unit

- ARMv7 MMU and LPAE compatibility
- LPAE enhancements to adapt to AArch64
- Supporting up to 48 bits of VA per TTBR
- Access permission checking
- Supporting up to 48 bits of IPA and PA spaces
- VMSAv8-64 address translation system
- Memory translation granule size
- Descriptor page table organization, descriptor format
- Hierarchical control of Secure or Non-secure memory accesses

## The ARMv8 Security Model

- Compatibility with ARMv7
- Security model when EL3 is using AArch64
- Trapping to EL3 using AArch64
- Re-entrant mode
- Secure exception management, trapping
- Asynchronous exception routing and control

## Virtualization

- New hypervisor privilege level on non-secure side
- Re-entrant mode
- Virtualization Extension Effect on MMU
  - Second stage MMU
  - I/O MMU
  - Managing external masters programmed by the guest OS without an I/O MMU
- Emulation support
- Hypervisor exception management, trapping
- Asynchronous exception routing and control
- Resource management
- Virtualization modes

- Para virtualization versus full virtualization
- Separation kernels
- Partitioning kernels
- Operating-system virtualization (containers)
- Existing hypervisors (Xen, KVM, ...)