Cortex-R8 Software Development

Course Description

Cortex-R8 software development is a four days ARM official course.
The course goes into great depth and provides all necessary know-how to develop software for systems based on Cortex-R8 processor.

The course covers the processor architecture, memory ordering, memory protection unit (MPU), caches and TCMs, Assembler language, synchronization, barriers, debug, power management, C for ARM and exception handling.

At the end of the course the participant will receive a certificate from ARM.

Course Duration

4 days

P.O.B. 803 Kfar Saba 441080 ISRAEL | P +972-52-5816791 | F +972-77-4702742 | ContactUs@HandsOnTraining.co.il | HandsOnTraining.co.il

Goals

1. Become familiar with ARMv7 architecture
2. Become familiar with Cortex-R8 architecture
3. Become familiar with ARM instruction sets
4. Understand Caches and TCMs structures and maintenance
5. Be able to write assembler code for Cortex-R8
6. Implement synchronization processes using mutex/semaphore
7. Be able to add barriers instructions to control program flow
8. Be able to configure and use the MPU
9. Apply invasive and non-invasive debug techniques
10. Write an efficient C code for Cortex-R8 processor
11. Be familiar with ARM tools for Cortex-R processors
12. Manage Cortex-R8 power modes

Target Audience

Software engineers that would like developing software and BSP for platforms based on Cortex-R8 processor.

Prerequisites

- Computer architecture background
- C and Assembler
- Experience in developing embedded systems

Course Material

ARM official course book
ARM DS5 SDK

Agenda

Main Topics:

- ARM Processor Cores
- ARM System Design
- Introduction to the ARM Architecture
- ISA Assembly
- Exception Handling
- Software Engineer's Guide to the Cortex-R8
- Assembler Programming for ARM Processors
- Exception Handling
- ARM Caches and TCMs
- Using the MPU
- Synchronization
- Barriers
- C/C++ Compiler Hints & Tips
- Linker & Libraries Hints & Tips
- Programming the GIC
- Further Compiler/Linker Hints & Tips
- Embedded Software Development
- Cortex-R power management
- Debug and trace

Day #1

❖ Introduction to the ARM architecture

- Architecture Versions
  - Introduction to the ARM architecture
  - Development of the ARM architecture
  - ARM Cortex processors (A/R/M)

- Registers & Instruction Sets
  - Data sizes and instruction sets
  - The ARM register set
  - Program status register
  - ARM, Thumb. Thumb2, ThumbEE, Jazelle
  - Endianness
  - Assembler syntax examples
  - Floating point and NEON
  - AAPCS

- Exception Model
  - Processor modes
  - Banking of registers
  - Taking an exception
  - Vector table

- Memory Model
  - Memory model overview
  - Memory types (Normal/Device/Strongly Ordered)
  - Memory hierarchy example
  - Data alignment

- Coprocessors
  - Coprocessors overview
  - CP15 example
  - PMU

- Architecture Extensions
  - TrustZone
  - Virtualization
  - Jazelle

❖ ARMv7-R ISA Overview

- ARM Assembler File Syntax
- Load/Store Instructions
  - Single/Double register data transfer
  - Addressing memory

- Pre and post-indexed addressing
- Multiple register data transfer

- Data Processing Instructions
  - Arithmetic, logical, move instructions
  - Shift/Rotate operations
  - The flexible second operand
  - Instructions for loading constants
  - Multiply/Divide
  - Bit manipulation instructions
  - Byte reversal

- Flow Control Instructions
  - Branch instructions
  - Interworking
  - Compare and branch if Zero
  - Condition codes and flags
  - If-Then instruction
  - Supervisor call instruction (SVC)

- Miscellaneous Instructions
  - Coprocessor instructions
  - PSR access
  - Breakpoint instruction (BKPT)
  - Wait for interrupt instruction (WFI)
  - NOP instruction
  - Wait for event & send event instructions (WFE & SEV)

- DSP Instructions
  - SIMD
  - Saturated maths and CLZ
  - Data packing/unpacking

❖ **Lab #1: ARM Assembler Programming Workbook**

❖ **Cortex-R8 Architecture**

- System Engineer's Guide to the Cortex-R8
  - Cortex-R8 architecture overview
  - Cortex-R8 specification and configuration
  - Major enhancements from the Cortex-R7 processor
  - Cortex-R8 pipeline
  - Instruction prefetch and branch prediction
  - Fast deterministic real-time response
  - Floating point unit (FPU)
  - Flexible memory with AXI-mapped SRAM

- L1 memory system
- Cache, TCM and LLRAM
- Real-time memory architecture
- Generic Interrupt Controller (GIC) and peripherals
- AXI-slave port to TCM
- Low-Latency Peripheral Port (LLPP)
- Fast Path Peripheral Ports (FPPP)
- Accelerator Coherency Port (ACP)
- Symmetric multi-processing
- Asymmetric multi-processing
- Quality of Service (QoS)
- Snoop Control Unit (SCU)
- Error management
- Memory reconstruction port
- Modes of operations (split, locked modes)
- Performance Monitoring Unit (PMU)
- CoreSight SoC Debug & Trace
- Power management
- Timers & Watchdogs

Lab #2: Measuring Code Performance using the PMU

Day #2

❖ **Exception Handling in Details**

➢ Introduction
- Exception handling process
- The ARM register set and modes
- Exception priorities
- Vector table
- Link register adjustments
- Returning from exceptions
- Exception state & Endianness
- Non-makable fast interrupt
- Low latency interrupt

➢ Interrupts
- Interrupt & interrupt handler example
- Interrupt pre-emption
- Issues with re-enabling interrupts
- Change processor state (CPS) instruction
- Stack issues

- Nested interrupt example
- FIQ vs IRQ
- Interrupt controllers

- Abort Handlers
  - Prefetch and data aborts
  - Data abort types (internal/external, precise/imprecise)
  - Identifying the abort source
  - Example data abort handler

- SVC Handlers
  - What are SVC used for?
  - Example SVC handler
  - Example SVC usage in an OS

- Undef Handlers
  - Undefined instruction
  - Example Undef handler

- Reset Handlers

❖ **Memory Structure**

- ARMv7-R Caches and TCMs
  - Cache basics
  - Caches on ARM processors
  - L1 data cache policies
  - Inner and Outer cache policies
  - Write back and write through
  - L1 memory system buffers
  - Tightly Coupled Memory (TCM)
  - TCM configuration
  - ECC and parity schemes
  - Optimization considerations
  - Cache coherency operations
  - Cache core optimizations
  - Point of Unification (PoU) and Point of Coherency (PoC)
  - Cache considerations
  - Using the PMU
  - Snoop Control Unit (SCU)
  - Hardware coherence management
  - Enabling coherency management
  - Accelerator Coherency Port (ACP)
  - Address filtering
  - Maintenance operations broadcast

P.O.B. 803 Kfar Saba 441080 ISRAEL | P +972-52-5816791 | F +972-77-4702742 | ContactUs@HandsOnTraining.co.il | HandsOnTraining.co.il

❖ **Memory Protection Unit (MPU)**

- Using the MPU
  - Why do we need memory management?
  - Access permissions
  - Memory types
  - Types & attributes
  - Instruction accesses
  - Memory Protection Unit (MPU) overview
  - Protection regions
  - When the MPU is disabled
  - Enhancements in Cortex-R8

Lab #3: Exception: Stack Overflow Protection using MPU

❖ **Using Barriers**

- Understanding Barriers
  - Memory model and access order
  - Barriers overview
  - Data barriers
    - DMB vs DSB
    - DMB in details
    - DSB in details
  - Mail box example
  - Speculation across barriers
  - Instruction barriers
    - ISB in details
  - CP15 example
  - Self-modifying code
  - Compiler barriers

❖ **Synchronization Techniques**

- Synchronization
  - The need for atomicity
  - Critical sections
  - LDREX and STREX instructions
  - Multi-thread mutex example
  - Coherent and non-coherent multi-core
  - Synchronization in a cluster
  - Memory attributes
  - Context switching

Day #3

❖ C/C++ Compiler Hints & Tips

- C/C++ Compiler Hints & Tips
  - Basic compilation
    - Source language modes of the compiler
    - Variables types supported
    - Compiler Optimization levels & debug view
    - Selecting an architecture or processor
  - Compiler optimizations
    - Idiom recognition
    - Tail-call optimization
    - Loop termination
    - Loop unrolling
    - Branch target optimization
    - Inline functions
    - Multifile compilation
- Writing C for ARM
  - Parameter passing
    - Parameter passing
    - Passing more than 4 parameters
    - Parameter alignment
  - Floating point linkage
    - HW and SW floating point linkage
    - Floating point linkage example
  - Alignment
    - Global data layout
    - Unaligned accesses
    - Packing and alignment of structures
    - Alignment of pointers
    - Optimization of memcpy()
  - Coding considerations
    - Size of local variables
    - Global & static variables
    - Global data in memory
    - Division
    - Division by compile-time constants
    - Modulo arithmetic
    - Base pointer optimization

P.O.B. 803 Kfar Saba 441080 ISRAEL | P +972-52-5816791 | F +972-77-4702742 | ContactUs@HandsOnTraining.co.il | HandsOnTraining.co.il

- Using “volatile”
- Conditional branch optimization

❖ **Lab #4: ARM Compiler Hints & Tips Workbook**

❖ **Linker & Libraries Hints & Tips**

➢ Linker Basics
  - o What does a linker do?
  - o How does the linker know what to do?
  - o Object file structure
  - o Library structure
  - o Scatter-loading

➢ System & User Libraries
  - o Libraries versus object files
  - o Linker library searching
  - o Creating and maintaining libraries

➢ Veneers & Interworking
  - o Dealing with branches
  - o Linker generated veneers
  - o Veneer types
  - o Minimizing the number of veneers

➢ Linker Optimizations & Diagnostics
  - o Unused section elimination
  - o RW data compression
  - o Small function inlining
  - o Useful linker diagnostics

➢ ARM Supplied Libraries
  - o ARM compiler standard libraries
  - o Microlib

❖ **Lab #5: ARM Linker Hints & Tips Workbook**

❖ **Further Compiler/Linker Hints & Tips**

➢ Mixing C/C++ and Assembler
  - o Register usage revisited
  - o Mixing C and assembly
  - o Calling assembly from C/C++
  - o Intrinsics
  - o Embedded assembler
  - o Inline assembler

- Stack Issues
  - Overview
  - Protecting and measuring stack usage
  - –callgraph example

- VFP
  - Floating point
  - Floating point and linkage

- Advanced Building Facilities
  - Multifile compilation
  - Linker feedback
  - Linking for specific target
  - Debug issues & build time

❖ **Lab #6: ScatterLoading Workbook**

Day #4

❖ **Generic Interrupt Controller Programming**

- GIC Overview
  - GIC architecture
  - Sources of interrupt (SGI, PPI, SPI)

- Distributor and CPU Interfaces
  - Register interfaces
  - Distributor interface
  - CPU interface
  - Programming guidelines

- How to Enable and Configure Interrupts
  - Enabling the IC
  - Interrupt configuration

- How to Handle Interrupts
  - Interrupt states
  - Taking an interrupt
  - Which CPU services an SPI?
  - Priority mask register
  - Interrupt priority registers
  - Pre-emption
  - Nesting interrupts

- How to Send Software Interrupts

- SGI capability
- Sending an SGI
- Receiving a SGI

- Security Extensions
  - Group 0 and Group 1
  - Acknowledging interrupts
  - Priority and banking

- Interrupts IDs on Cortex-R8

❖ **Lab #7: Nested Interrupts using the GIC**

❖ **Debug Overview**

- Debug
  - Types of debug
  - Invasive debug
  - Debug infrastructure and CoreSight overview
  - How do I access the debug logic?
  - Debug events
  - Halt vs. Monitor mode debugging
  - Viewing memory
  - Debugger impact on caches
  - Vector catch
  - Instruction breakpoint types
  - Embedded cross trigger- CTI
  - Debugger semi-hosting support
  - Non-invasive debug (PMU and trace)
  - Performance monitoring hardware
  - PMU configuration
  - Results analysis
  - CoreSight trace system
  - Other trace resources

❖ **Software Power Management for Cortex-R8**

- Cortex-R8 Power Management
  - Processor power consumption
    - Power components
    - Example power contributions
    - Power reduction
  - Power modes
    - Standby mode
    - Shutdown mode
    - Dormant mode
  - Use cases

- SoC power modes
- Entering low power modes
- Exiting shutdown/dormant mode