[redacted]

This course covers and explains the implementation of the RISC-V CPU

Objective

- Understand the basics of the RISC-V architecture and instruction set.
- Develop proficiency in RISC-V C Programming and be able to write, compile, and run RISC-V C code.
- Learn how to handle interrupts and exceptions in RISC-V.
- Understand the concepts of RISC-V hardware and system design, specifically on FPGA and embedded systems

A more detailed course description is available on request at training@ac6-training.com

Course Environment

- Theoretical course
  - PDF course material (in English).
  - Course dispensed using the Teams video-conferencing system.
  - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance through the Teams video-conferencing system.
- Practical activities
  - Practical activities represent from 40% to 50% of course duration.
  - Code examples, exercises and solutions
  - One Online Linux PC per trainee for the practical activities.
  - The trainer has access to trainees' Online PCs for technical and pedagogical assistance.
  - Eclipse environment and GCC compiler.
  - QEMU Emulated board or physical board connected to the online PC (depending on the course).
  - Some Labs may be completed between sessions and are checked by the trainer on the next session.
- Downloadable preconfigured virtual machine for post-course practical activities
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

Prerequisites

- Familiarity with computer architecture
- Programming skills: Some programming experience, particularly in C or assembly
- Knowledge of digital logic: Understanding of digital logic and basic concepts of computer design would be beneficial for understanding RISC-V CPU implementation and FPGA design
- Basic understanding of operating systems: Familiarity with operating system concepts such as process management, memory management, and interrupts
- The course may use Linux-based development tools and environments

Target Audience

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

Evaluation modalities

- The prerequisites indicated above are assessed before the training by the technical supervision of the traineein his company, or by the trainee himself in the exceptional case of an individual trainee.
- Trainee progress is assessed in two different ways, depending on the course:

  - For courses lending themselves to practical exercises, the results of the exercises are checked by the trainer while, if necessary, helping trainees to carry them out by providing additional details.
  - Quizzes are offered at the end of sections that do not include practical exercises to verifythat the trainees have assimilated the points presented
- At the end of the training, each trainee receives a certificate attesting that they have successfully completed the course.
  - In the event of a problem, discovered during the course, due to a lack of prerequisites by the trainee a different or additional training is offered to them, generally to reinforce their prerequisites,in agreement with their company manager if applicable.

Plan

First Session

Introduction to RISC-V

- Overview of RISC-V
  - What is RISC-V and why is it important?
  - History and development of RISC-V
  - RISC-V architecture and instruction set
  - RISC-V implementations and applications
- RISC-V ISA Overview
  - Instruction format
  - Instruction set encoding
  - Privileged architecture
  - Vector instructions
  - Compressed instructions

Exercise: *Setting up a RISC-V development environment and running a "Hello World" program on a RISC-V emulator*

RISC-V CPU Implementation

- RISC-V 32-bit CPU implementation
  - RISC-V 32-bit instruction set
  - RISC-V 32-bit register set
  - RISC-V 32-bit pipeline
- RISC-V 64-bit CPU implementation
  - RISC-V 64-bit instruction set
  - RISC-V 64-bit register set
  - RISC-V 64-bit pipeline

Exercise: *CPU Implementation*

Second Session

RISC-V Memory Management

- Introduction to RISC-V memory management
  - Memory management unit
  - Virtual memory
  - Address translation
- Memory-mapped I/O in RISC-V
  - MMIO interface
  - Device drivers
- Virtual memory and address translation in RISC-V
  - Page tables
  - Translation lookaside buffer

RISC-V Interrupt and Exception Handling

- Introduction to RISC-V interrupts and exception handling
  - Interrupts and exceptions
  - Interrupt handling
- Implementing interrupt handlers in RISC-V assembly and C
  - Interrupt service routines
  - Exception handling
- Handling exceptions and errors in RISC-V
  - Exception vectors
  - Trap handling

Exercise: *Interrupt and Exception Handling*

RISC-V C Programming and Debugging

- Introduction to RISC-V C programming
  - Setting up the C development environment
  - Writing and compiling RISC-V C code
- Debugging and testing C code
  - GDB and OpenOCD
  - Trace

Exercise: *C programming and debugging*

Third Session

RISC-V Optimization

- Introduction to RISC-V performance optimization
  - Understanding performance metrics
  - Identifying performance bottlenecks
- Profiling and benchmarking RISC-V code
  - Using performance counters
  - Analyzing performance data
- Optimizing RISC-V code for performance
  - Instruction scheduling
  - Loop optimization
  - Register allocation
  - Memory optimization

Exercise: *Optimizing RISC-V Code*

RISC-V Optimization for FPGA and Embedded Systems

- Introduction to RISC-V on FPGA
  - Overview of FPGA technology
  - RISC-V on FPGA: benefits and challenges
- Synthesis and Implementation
  - Synthesis flow
  - Place and route
  - Power and performance optimization
- Designing RISC-V systems with FPGA
  - SoC design
  - Peripherals and interfaces
  - Interrupts and exception handling
- RISC-V on embedded systems and IoT applications
  - Applications and use-cases
  - Memory and power constraints
  - Security and privacy concerns

Exercise: *Implementing a RISC-V system on an FPGA development board*

Renseignements pratiques

Inquiry : 18 hours