RV1 - Architecture RISC-V

Ce cours détaille l'implémentation des CPU RISC-V

Objectifs

- Comprendre les bases de l'architecture et du jeu d'instructions RISC-V.
- Acquérir des connaissances sur le jeu d'instruction RISC-V.
- Développer la maîtrise de la programmation RISC-V C et être capable d'écrire, de compiler et d'exécuter du code RISC-V C.
- Apprendre à gérer les interruptions et les exceptions dans RISC-V.
- Comprendre les concepts de conception matérielle et système RISC-V, en particulier sur les FPGA et les systèmes embarqués

Une description plus détaillée est disponible sur demande à training@ac6-training.com

Environnement de cours

- Cours théorique
  - Support de cours au format PDF (en anglais)
  - Cours dispensé via le système de visioconférence Teams
  - Le formateur répond aux questions des stagiaires en direct pendant la formation et fournit une assistance technique et pédagogique
- Activités pratiques
  - Un PC Linux en ligne par stagiaire pour les 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.
  - Certains exercices utilisent QEMU pour émuler un CPU RISC-V pour exécuter du code assembleur ou C
  - Exemples de code, exercices et solutions
  - Le formateur a accès aux PC en ligne des stagiaires pour l'assistance technique et pédagogique
  - Certains travaux pratiques peuvent être réalisés entre les sessions et sont vérifiés par le formateur lors de la session suivante
  - Le cours fournit des descriptions et des tutoriels pour les sections pratiques basées sur QEMU
- Une machine virtuelle préconfigurée téléchargeable pour les activités pratiques après le cours est aussi fournie
- 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

Prérequis

- Connaissance de l'architecture informatique
- Compétences en programmation : Une certaine expérience en programmation, en particulier en C ou en assembleur
- Connaissance de la logique numérique : la compréhension de la logique numérique et des concepts de base de la conception informatique serait bénéfique pour comprendre la mise en Œuvre du processeur RISC-V sur FPGA
- Compréhension de base des systèmes d'exploitation : connaissance des concepts des systèmes d'exploitation tels que la gestion des processus, la gestion de la mémoire et les interruptions
- Le cours peut utiliser des outils et des environnements de développement basés sur Linux

Audience visée

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

Course Outline

Premier Jour

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

Deuxième Jour

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

Troisième Jour

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