RISC-V For Heterogeneous Computing

Zhipeng Huang, Huawei
Justin Cormack, Docker
Recap from Zurich Workshop

Cloud Management (OpenStack, Kubernetes, etc...)

RISC-V Core Capabilities (topology, socket closeness, affinity, power, ...)

RISC-V Core Based Accelerator
Heterogeneous Computing at Edge/Urban
Heterogeneous Computing at Edge/Urban

Edge

- Smart Traffic
- Smart Factory
- Smart Building
- Smart Vehicles

Urban

- Smart Home
- Mobile Office
- Fitness
- Travel

Far

Close
Heterogeneous Computing at Edge/Urban Open Source Ecosystem

- KubeEdge
- Docker
- RISC-V LLVM Toolchain
- LiteOS
- RISC-V LLVM Toolchain
- OpenArk Compiler

RISC-V As Common Infrastructure For Edge and Urban Computing

Application Interoperability and portability
KubeEdge Overview

https://github.com/kubeedge/kubeedge
KubeEdge RISC-V Support

- KubeEdge RISC-V Support mostly depends on the EdgeCore part.
- Current Docker RISC-V build could solve most of the problem, with the exception of syscall and MQTT driver support.
- ONNX Runtime RISC-V build support would be another great to have.
- Thanks Carlos for all the hard work at https://github.com/carlosedp/riscv-bringup
LiteOS: One Lightweight Kernel & Multiple Frameworks For Industrial IoT Device

https://github.com/LiteOS/LiteOS

<table>
<thead>
<tr>
<th>Connectivity Framework</th>
<th>Sensor Framework</th>
<th>Security Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connectivity and auto-networking of multi-protocol devices</td>
<td>Unified sensor management</td>
<td>Security protection for terminals</td>
</tr>
<tr>
<td>Application profile</td>
<td>Sensing algorithm library</td>
<td>API authentication</td>
</tr>
<tr>
<td>Auto-networking APIs</td>
<td>Sensor management</td>
<td>Bidirectional device authentication</td>
</tr>
<tr>
<td>Device-to-Cloud protocol HTTP/CoAP/MQTT/LWM2M</td>
<td>Driver management</td>
<td>Encryption algorithm library</td>
</tr>
<tr>
<td>Network protocol stack uIP/lwIP/RPL</td>
<td>Communication protocol BLE/WiFi/6LowPAN/Zigbee/PLC/NB-IoT</td>
<td></td>
</tr>
</tbody>
</table>

Kernel Functions

- RISC-V
- ARM Cortex-M0, Cortex-M3, Cortex-M4, Cortex-M7
- ARM Cortex-A7, Cortex-A17, Cortex-A53
- ARM11
- DSP
LiteOS: RISC-V Support

- POSIX interface (Libc/Libm/STL C/C++ standard library)
- Task management
- Memory management
- Interrupt management
- SafeArea
- Hardware abstraction layer
- Chip drivers
- Drivers of peripheral devices

Lightweight Kernel
- Scalable (minimum 6 KB)
- Microampere-level power consumption
- Microsecond-level responding time
- Multiple chip architectures
- Standard interfaces

MCU Startup
Task Switch
Tick
Interrupt Handling
Exceptional Handling
RISC-V
OpenArkCompiler Overview

Compiler

Front-End
- Jbc2MapleIR

Middle-End
- SSA Building
- RC Insertion
- RC Tuning
- Escape Analysis
- Copy Propagation
- De-virtualization
- PRE Optimization

Back-End
- Memory layout
- Instruction Selection
- Control Flow Optimization
- EBO Optimization
- Peephole Optimization
- Register Allocation

MAPLE IR SPEC

Runtime

- RC
- Tracing GC
- Thread Modeling
- JNI
- Reflection
- Anomaly Handling
- Allocator
- String
- OS Adaptor
- Object Modeling

https://gitee.com/harmonyos/OpenArkCompiler
RISC-V Support in OpenArkCompiler

Source Code

Language Frontends

Mid Level Optimizer

Targeted Backends

ARM  X86  ...  RISCV

Runtime

TODO List:

- Instruction Selection
- Frame Lowering
- Calling Conventions
- Register Allocation
- Instruction Scheduling
- Optimizations
- ...

RISC-V Support in OpenArkCompiler
RISC-V Toolchain Development

**Work In Progress**

- Vector Extension Based On LLVM
  - Assembly Support (current stage) => Intrinsic Function Support => Vectorization
  - [https://github.com/isrc-cas/rvv-llvm](https://github.com/isrc-cas/rvv-llvm)

- Eternal Balance (Benchmark CI)
  - [https://github.com/isrc-cas/Eternal-Balance](https://github.com/isrc-cas/Eternal-Balance)

- Tiger Compiler for RISC-V

- LLVM CodeGen Analysis
  - Register Allocation
  - Instruction Scheduling
  - Opportunistic Scheduling
  - Soft Pipeline
  - Target Description Model

**Future Work**

- RISC-V Code Size Optimization
- RISC-V backend micro-architecture optimizations
- Linker Relax Optimization
- Linker Time Optimization
- Exchange between LLVM IR and OpenArkCompiler MAPLE IR
- ps abi
- Code Obfuscation

*Thanks to PLCT Lab team effort*