A unique processor architecture meeting LLVM IR and the IoT

Dávid Juhász
david.juhasz@imsystech.com
Compiling with LLVM

High-Level Programming Language

LLVM Front-end

LLVM Assembly / IR

LLVM Back-end

Your Typical Target ISA

LLVM Middle-end

LLVM
Improving Efficiency by Lifting Target

High-Level Programming Language

LLVM Front-end

LLVM Assembly / IR

LLVM Back-end

Imsys ISA for LLVM

Your Typical Target ISA

LLVM

Big gap

Complex translation

Reducing the gap
Agenda

1. Imsys Company and the IoT
2. Imsys Lean Processing Technology
3. Imsys ISA for LLVM
Networked Embedded Controllers
Imsys EMBLA
Single Controller Solution for the IoT
High-Level Software Platform

C/C++  Java  Application Code

LLVM  Java  eclipse

Executable Binary

Integrated Development Environment
Improving Efficiency by Reducing the ISA Gap

High-Level Programming Language

LLVM Front-end

LLVM Assembly / IR

LLVM Back-end

Your Typical Target ISA

LLVM Middle-end

Imsys ISA for LLVM

Reducing the gap

Big gap

Complex translation
Abstraction Layers

Application Code

Assembly/C/C++

JAVA

ISAL

ISAJ

Imsys Instruction Set Support

Imsys Processor Core
A Forgotten Layer of Abstraction
The Processor Architect’s Best Friend

Operation-Oriented Hardware Architecture
Microcode, what is it good for?

- Complete Deterministic Control
- Minimal Hardware
- Maximum Efficiency
- Flexibility
- Soft-Reconfigurability
Abstraction Layers Revisited

Application Code

ISAL

Imsys Instruction Set Support

Imsys Processor Core

Software

Microcode

Hardware
A Unique Balance between Hardware and Software

Application

Compiler

Microcode

Balanced Rich ISA

Domain-specific Operations

Operation-Oriented Hardware Architecture
Improving Efficiency by Matching LLVM Assembly

High-Level Programming Language

LLVM Front-end

LLVM Assembly / IR

LLVM Back-end

Imsys ISA for LLVM

Your Typical Target ISA
Matching LLVM Assembly

Simple, efficient, general LLVM

LLVM Assembly

LLVM back-end for ISAL

Semantically matching instructions

ISAL

LLVM middle-end

Direct use of optimizations
Meeting a Constrained Reality

<table>
<thead>
<tr>
<th></th>
<th>LLVM Assembly</th>
<th>ISAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations</td>
<td>Instructions &amp; intrinsic functions</td>
<td>Additional system operations</td>
</tr>
<tr>
<td>Single Value Types</td>
<td>Virtually unlimited</td>
<td>Set of integer, floating point, pointer, and vector types</td>
</tr>
<tr>
<td>Registers</td>
<td>Unlimited</td>
<td>Register windows</td>
</tr>
<tr>
<td>Arguments</td>
<td>Source and destination registers</td>
<td>Registers with support of accumulating in source registers, immediate values</td>
</tr>
<tr>
<td>Binary Representation</td>
<td>Bitcode</td>
<td>Custom dense binary coding</td>
</tr>
</tbody>
</table>
Optimized Operation Sequences

a := a + b

\[
\text{add} \, a \, a \, b
\]

\[
\text{add.update} \, a \, b
\]

Accumulating in source register

 opcode dst src1 src2

a := a + 42

\[
\text{move} \, b \, 42
\]

\[
\text{add} \, a \, a \, b
\]

Immediate values

 opcode src1 src2

\[
\text{add.immediate} \, a \, a \, 42
\]

\[
\text{add.update.immediate} \, a \, 42
\]
Optimized Binary Representation

Maximize Code Density
Optimize Microcode Size
Minimize Execution Time

Variable-length Instructions

<table>
<thead>
<tr>
<th>Instruction Lengths (Bytes)</th>
<th>Number of Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>198</td>
</tr>
<tr>
<td>3</td>
<td>465</td>
</tr>
<tr>
<td>4</td>
<td>258</td>
</tr>
<tr>
<td>5</td>
<td>61</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
</tr>
</tbody>
</table>
ARM Cortex requires 35%+ more, Intel 80%+ more program memory.
LLVM Assembly

LLVM back-end for ISAL

ISAL

Imsys Firmware

Imsys Processor Core

Imsys Lean Processing Technology

Imsys ISA for LLVM

Imsys EMBLA with ISAL

Leaner is Meaner

Dávid Juhász
david.juhasz@imsystech.com

FOSDEM 2018 LLVM devroom