Alternative languages for safe and secure RISC-V programming

Fabien Chouteau

Embedded Software Engineer at AdaCore

Twitter: @DesChips
GitHub: Fabien-Chouteau
Hackaday.io: Fabien.C
What do I mean by “alternative”?
What do I mean by “alternative”?
Functional Safety
Specifications

type Servo_Angle is new Float range -90.0 .. 90.0
-- Servo rotation angle in degree

procedure Set.Angle (Angle : Servo_Angle);
-- Set desired angle for the servo motor
Contracts

type Stack is tagged private;

function Empty (S : Stack) return Boolean;
function Full (S : Stack) return Boolean;

procedure Push (S : in out Stack; Val : Integer)
  with Pre => not S.Full,
       Post => not S.Empty;

procedure Pop (S : in out Stack; Val : out Integer)
  with Pre => not S.Empty,
       Post => not S.Full;
Checks

- At run-time
  - Checks inserted in the code
  - For debug or testing

- At Compile time
  - Compiler
  - Static analyzer
  - Formal verification (SPARK)
-- High level view of the Sense field

type Pin_Sense is
  (Disabled,
   High,
   Low)
  with Size => 2;

-- Hardware representation of the Sense field

for Pin_Sense use
  (Disabled => 0,
   High    => 2,
   Low     => 3);
-- High level view of the register

type IO_Register is record
    Reserved_A : UInt4;
    SENSE : Pin_Sense;
    Reserved_B : UInt2;
end record with Size => 32;

-- Hardware representation of the register

for IO_Register use record
    Reserved_A at 0 range 0 .. 3;
    SENSE at 0 range 4 .. 5;
    Reserved_B at 0 range 6 .. 7;
end record;
#define SENSE_MASK (0x30)
#define SENSE_POS (4)
#define SENSE_DISABLED (0)
#define SENSE_HIGH (2)
#define SENSE_LOW (3)

uint8_t *register = 0x80000100;

// Clear Sense field
*register &= ~SENSE_MASK;

// Set sense value
*register |= SENSE_DISABLED << SENSE_POS;
Hardware mapping

Register : IO_Register

with Address => 16#8000_0100#;

Register.SENSE := Disabled;
<field>
  <name>SENSE</name>
  <description>Pin sensing mechanism.</description>
  <lsb>16</lsb>  <msb>17</msb>
  <enumeratedValues>
    <enumeratedValue>
      <name>Disabled</name>
      <description>Disabled.</description>
      <value>0x00</value>
    </enumeratedValue>
  </enumeratedValues>
[...]
with Interfaces.C; use Interfaces.C;

function My_C_Function (A : int) return int
  with Pre => A /= 0;
pragma Import (C, My_C_Function, "my_c_function");

function My_Ada_Function (A : int) return int;
pragma Export (C, My_Ada_Function, "my_ada_function");
Getting started on RISC-V
Hardware
Download and install the tools: adacore.com/community

Download GNAT Community Edition
For free software developers, hobbyists, and students.

<table>
<thead>
<tr>
<th>Platform</th>
<th>GNAT Community</th>
<th>SHA-1</th>
<th>Size</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RISC-V ELF</strong> (hosted on linux64)**</td>
<td><strong>README.txt</strong></td>
<td>f48a0f0add58f160f0d35023ee358e7d3be4f0d8c</td>
<td>2.1 KiB</td>
<td>Jul 10 2018</td>
</tr>
<tr>
<td></td>
<td><strong>gnat-community-2018-20180524-riscv32-elf-linux64-bin</strong></td>
<td>1e2dfca066b34247f26c1e4387c8b25f18e6c40aef8</td>
<td>137.2 MiB</td>
<td>May 28 2018</td>
</tr>
<tr>
<td><strong>x86-64 GNU Linux</strong> (64 bits)</td>
<td><strong>README.txt</strong></td>
<td>148a0f0add58f160f0d35023ee358e7d3be4f0d8c</td>
<td>2.1 KiB</td>
<td>Jul 10 2018</td>
</tr>
<tr>
<td></td>
<td><strong>gnat-community-2018-20180528-x86_64-linux-bin</strong></td>
<td>1060eb65596d40f340f672441a84f5cb0077ffcc</td>
<td>430.8 MiB</td>
<td>May 29 2018</td>
</tr>
<tr>
<td><strong>ARM ELF</strong> (hosted on linux64)**</td>
<td><strong>README.txt</strong></td>
<td>f48a0f0add58f160f0d35023ee358e7d3be4f0d8c</td>
<td>2.1 KiB</td>
<td>Jul 10 2018</td>
</tr>
<tr>
<td></td>
<td><strong>gnat-community-2018-20180524-arm-elf-linux64-bin</strong></td>
<td>148a0f0add58f160f0d35023ee358e7d3be4f0d8c</td>
<td>199.9 MiB</td>
<td>May 28 2018</td>
</tr>
</tbody>
</table>
Download Ada Drivers Library

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

https://github.com/AdaCore/Ada_Drivers_Library

Download ZIP
Embedded programming competition
Open to everyone
$8000 in prizes
Keep the door open
What as already been done

- Open specs and documentation
- RISC-V support in open-source tools:
  - Compilers (GCC, LLVM)
  - Debuggers (Gdb, openocd)
  - Simulators (QEMU)
Challenges

- Complexity of extension combinations
  RV(32|64|128) I M A C B [F|D|Q] …
- Deviation from the standard
- Custom/proprietary extensions
Do we need to go beyond SVD?

- Registers ✓
- Interrupts ✓
- CPU specs ?
- RAM and ROM banks ?
- Modular representation ?
- A mix between Device Tree and SVD?
- Tools that generate SVD from custom design ?
• learn.adacore.com: interactive learning website
• Competition: makewithada.org
• Twitter: @AdaProgrammers
• Reddit: r/ada