CS340400 Compiler Design
Homework 2 Bonus

RISC-V P Extension

RISC-V ISA Overview

- RISC-V is a family of related ISAs.
- A RISC-V ISA is defined as a base integer ISA + extensions.
  - Base ISA must be present in any implementation
  - Extensions are optional
- A RISC-V instruction-set variant can be described by concatenating the base integer prefix and the names of extensions.
  - E.g. “RV64IMAFD”

Source: Waterman, Andrew; Asanović, Krste (13 December 2019). "The RISC-V Instruction Set Manual, Volume I: Unprivileged ISA version 20191213" (PDF). RISC-V International.

RISC-V P Extension

- The RISC-V P extension (RVP) enhances the RISC-V CPU IP products with DSP capabilities that can run DSP applications with lower power and higher performance.
- The proposal of P extension is based on AndeStar$^{\text{TM}}$ V3 DSP ISA.

Digital Signal Processing (DSP)

- Digital signal processing algorithms perform mathematical operations on the digital representation of signals.
- Many DSP applications require low latency or power effiency constraints, which can be achieved by specialized ISA like RVP.
- Applications of DSP:
  - Audio signal processing
  - Digital image processing

RVP Instruction

- RVP instructions operate on the general purpose register (GPR) in the base ISA.
- There are three main types of instruction in RVP, which are SIMD[1], Partial-SIMD and Non-SIMD instructions, depending on the number of elements the instruction operates on.
- Some of the instructions are for RV64 only.

[1] SIMD: Single instruction, multiple data

RVP Instruction

- Example – 8-bit Signed Saturating Addition
  - Mnemonic: KADD8 rd, rs1, rs2
  - This add instruction will saturate to the range of $[-2^7, 2^7-1]$.
  - For instance, on RV32: rs1 = FF7F_FFFF_{16}, rs2 = 0101_0101_{16}
    - rd = (-1, 127, -1, -1) + (1, 1, 1, 1) = (0, 127, 0, 0) = 007F_0000_{16}
    - $7F_{16} + 01_{16}$ ($7F_{16} = 01111111_2$) = 127 + 1 = 127 (saturation)
    - $FF_{16} + 01_{16}$ ($FF_{16} = 11111111_2$) = -1 + 1 = 0

Homework 2 Bonus - Specification

Extension of C Language

- In homework 1, 2 and 3, we implement a compiler for a subset of C language.
- In the bonus problem of HW2, we will modified our compiler to support extension syntax that helps us generate RISC-V P extension instruction in HW3.

Extension of C Language – Types

- char4:
  - Declare an object with a width of 32 bits that stores 4x8-bit char elements as value.
- char8:
  - Declare an object with a width of 64 bits that stores 8x8-bit char elements as value.

Extension of C Language – Assignment Expression

- Implicit cast assignment:
  - char4 variable can be assigned to int variable and vice versa
  - char8 variables can be assigned to long variable and vice versa
  - E.g. char4 a,b; int c; a=b; a=c; c=a;
- Constant assignment:
  - Interpret the constant literal as the corresponding int/long
  - char4 a = 10; → a = 10_{10} = 0000_000A_{16}
  - char8 b = -1; → b = -1_{10} = $FFFF\_FFFF\_FFFF\_FFFF_{16}$
- Note:
  - The semantics defined above is for HW3 purpose, in HW2, we only need to take care of the syntax.

Extension of C Language – Arithmetic Expression

- SIMD 8-bit Saturating Addition & Subtraction Instruction:
  - char4 a = 1, b = -1; → a = 0000_0001_{16}, b = $FFFF\_FFFF_{16}$
  - a = a + b; → a = (0, 0, 0, 1) + (-1, -1, -1, -1) = $FFFF\_FF00_{16}$
- SIMD 8-bit Saturating Multiply Instruction:
  - char4 a = 1, b = -1;
  - a = a * b; → a = (0, 0, 0, 1) * (-1, -1, -1, -1) = 0000_00FF_{16}
- Note:
  - The semantics defined above is for HW3 purpose, in HW2, we only need to take care of the syntax.

Implement – Scalar Declaration/Expression

- Follows the specification in HW2 and extends the following rules:
  - Support scalar declaration for char4 and char8 types.
    - char4 a = 10; → <scalar_decl>char4a=<expr>10</expr>;</scalar_decl>
  - Support arithmetic expressions with +, -, *, (,), = operators for char4 and char8 types.
- Only a few modifications are required, but make sure you keeps the type information, which is important for HW3 bonus.

Grading Policies

- Total 10 bonus points:
  - Basic testcase: 5 points
  - Advanced testcase: 5 points
  - Note that if you scores over 100 in HW2, the additional points are added to the final score of the class.
- Basic testcase:
  - Will be released with its answer on the server.
- Advanced testcase:
  - Hidden testcase & requires most of the features in HW2.
- There is no golden parser for the bonus problem.

Reference

- **RISC-V Specifications:**
  - https://riscv.org/technical/specifications/
- **RISC-V “P” Extension Proposal:**
  - https://github.com/riscv/riscv-p-spec/blob/master/P-ext-proposal.adoc