中国科学院软件研究所学术年会’2020
暨计算机科学国家重点实验室开放周

学术论文

基于LLVM的RISC-V向量扩展支持

RISC-V Vector Extension Support in LLVM

SRC of 20th international symposium on Code Generation and Optimization
San Diego, CA, USA, February 22 – 26, 2020

Yin Zhang, Mingjie Xing, Yanjun Wu
Institute of Software Chinese Academy of Sciences Beijing, China
Corresponding: zhangyin2018@iscas.ac.cn

Introduction

As a modular ISA, RISC-V has been divided into an integer base with several standard extensions. One of these is the vector operation extension. Programming with these extensions requires the toolchain support. Currently, RISC-V vector extension support in LLVM is still in the early stage. We intend to fully support the RISC-V vector extension in LLVM including auto-vectorization in the future. So far, the assembly support for all the vector instructions and part of the intrinsic functions are completed. This paper is mainly about the assembly support we have implemented.

Implementation

LLVM uses TableGen to describe architecture features such as register and instruction information and generates backend related code automatically.

The general process of assembly implementation includes following parts:

a. Describe register information.

b. Describe instruction information.

a. Describe Register Information

Vector registers are described in the register information file of RISC-V backend (*RISCVRegisterInfo.td*). Both the register and the register classification should be described. They have inheritance relationships shown in **Fig.1**.

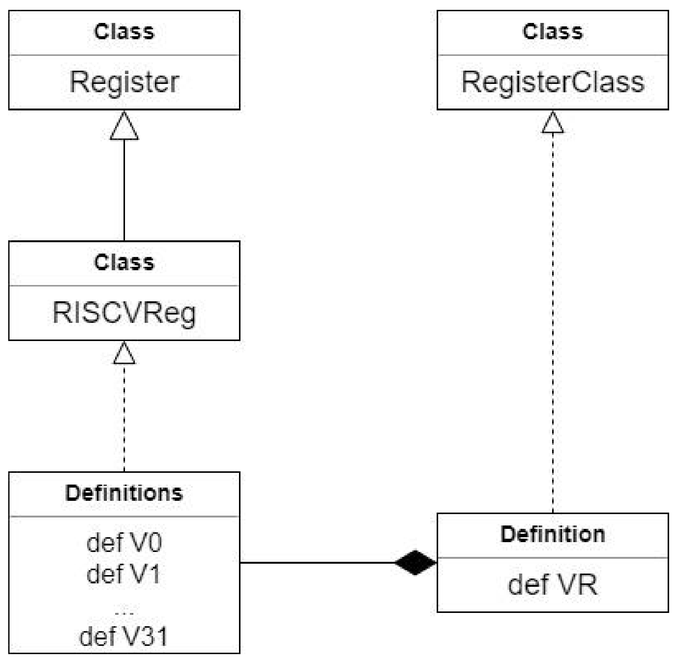

Figure 1: The Inheritance Relationship of Register Records

b. Describe Instruction Information

After defining the registers, we also describe vector instructions in RISC-V instruction format and information files (*RISCVInstrFormatV.td* and *RISCVInstrInfoV.td*). Instruction records have inheritance relationships shown in **fig.2**.

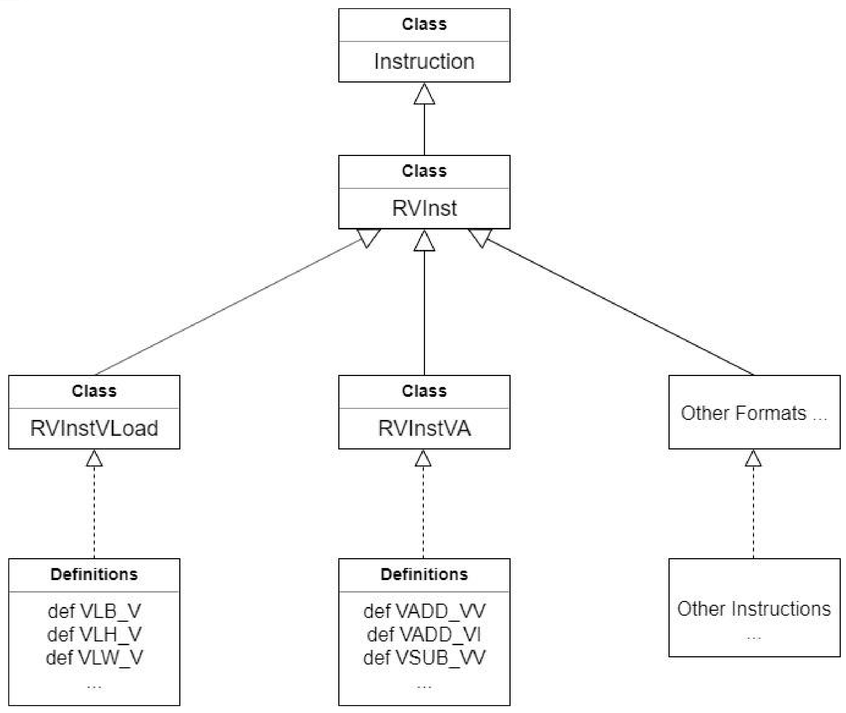

Figure 2: The Inheritance Relationship of Instruction Records

Conclusion

Our work supplies the deficiency of RISC-V vector extension in LLVM. Through this work, we find that adding new assembly instruction support in LLVM backend can be done mainly by writing target description. Developers should be familiar with the TableGen language and clearly understand the inheritance relationship of the backend features to be described. In addition, the testing infrastructure should be utilized to verify the correctness.