ISCAS 中国科学院软件研究所
Institute of Software Chinese Academy of Sciences
智能软件研究中心
Intelligent Software Research Center

LLVM中的RISC-V向量扩展支持

PLCT实验室　张尹

2019/12/15

CONTENT

目录

CONTENT

目录

1 背景介绍

LLVM中的RISC-V向量扩展支持

LLVM的RISC-V后端目前还没有对向量扩展的支持。

我们计划实现LLVM中的RISC-V向量扩展的支持。

目前我们已经基本完成了RISC-V向量扩展指令的汇编支持，后续会进一步对LLVM实现完整的RISC-V向量扩展支持。

本次报告主要阐述RISC-V向量扩展的汇编代码实现与后续向量扩展支持所需的相关工作。

1 背景介绍

RISC-V向量扩展简介

我们的汇编实现基于RISC-V向量扩展的稳定版本v0.7.1.

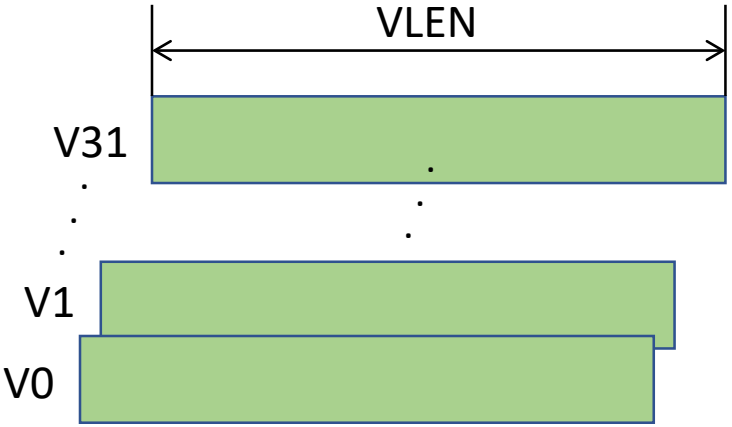
Base	Version	Status
RVWMO	2.0	**Ratified**
RV32I	**2.1**	**Ratified**
RV64I	**2.1**	**Ratified**
RV32E	*1.9*	*Draft*
RV128I	*1.7*	*Draft*
Extension	Version	Status
Zifencei	**2.0**	**Ratified**
Zicsr	**2.0**	**Ratified**
M	**2.0**	**Ratified**
A	*2.0*	Frozen
F	**2.2**	**Ratified**
D	**2.2**	**Ratified**
Q	**2.2**	**Ratified**
C	**2.0**	**Ratified**
Ztso	*0.1*	*Frozen*
Counters	*2.0*	*Draft*
L	*0.0*	*Draft*
B	*0.0*	*Draft*
J	*0.0*	*Draft*
T	*0.0*	*Draft*
P	*0.2*	*Draft*
V	*0.7*	*Draft*
N	*1.1*	*Draft*
Zam	*0.1*	*Draft*

模块化的RISC-V指令集

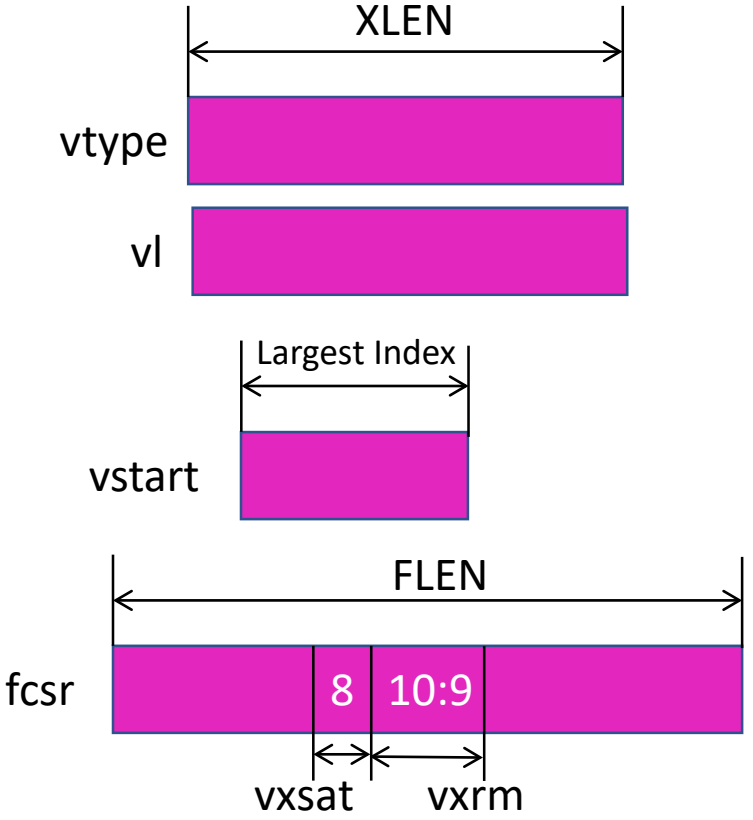
向量扩展“V”

1 背景介绍

RISC-V向量扩展寄存器



32个向量寄存器



5个控制与状态寄存器

1 背景介绍

RISC-V向量扩展指令

共662条向量指令

向量指令形式 ⟶

```
Format for Vector Load Instructions under LOAD-FP major opcode
31 29 28 26  25  24      20 19       15 14   12 11      7 6     0
 nf  | mop | vm |  lumop   |    rs1    | width |    vd   |0000111| VL*  unit-stride
 nf  | mop | vm |   rs2    |    rs1    | width |    vd   |0000111| VLS* strided
 nf  | mop | vm |   vs2    |    rs1    | width |    vd   |0000111| VLX* indexed
  3     3     1      5          5         3        5       7

Format for Vector Store Instructions under STORE-FP major opcode
31 29 28 26  25  24      20 19       15 14   12 11      7 6     0
 nf  | mop | vm |  sumop   |    rs1    | width |   vs3   |0100111| VS*  unit-stride
 nf  | mop | vm |   rs2    |    rs1    | width |   vs3   |0100111| VSS* strided
 nf  | mop | vm |   vs2    |    rs1    | width |   vs3   |0100111| VSX* indexed
  3     3     1      5          5         3        5       7
```

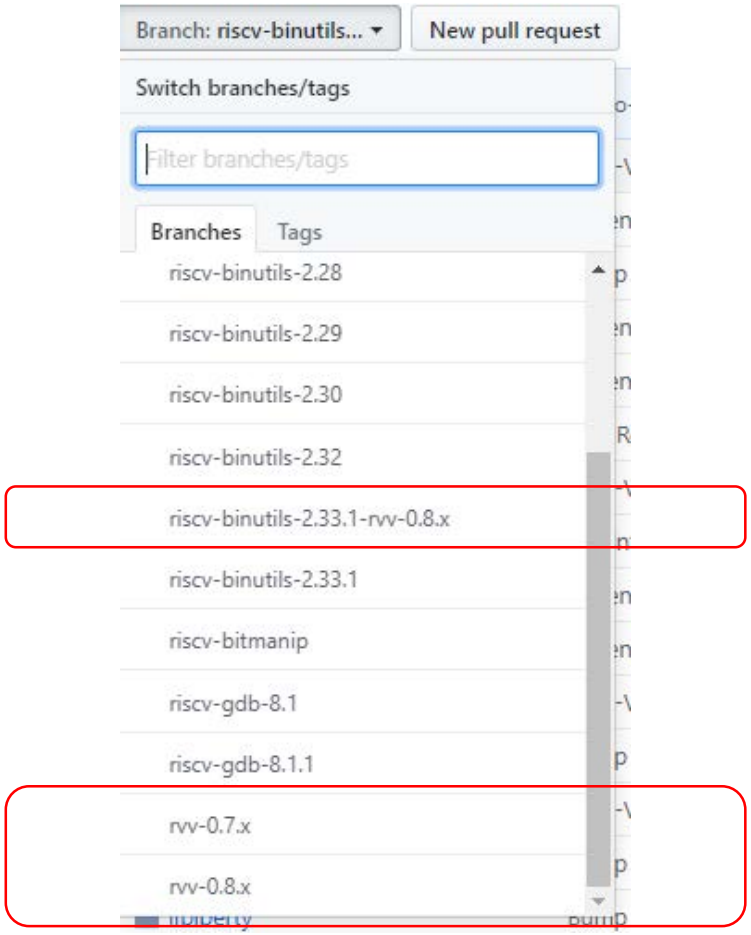
```
Format for Vector AMO Instructions under AMO major opcode
31    27 26  25  24      20 19       15 14   12 11      7 6     0
 amoop  |wd| vm |   vs2    |    rs1    | width | vs3/vd  |0101111| VAMO*
   5      1   1      5          5         3        5       7
```

```
Formats for Vector Arithmetic Instructions under OP-V major opcode

31       26  25   24      20 19      15 14   12 11      7 6     0
  funct6   | vm  |   vs2    |    vs1    | 0 0 0 |    vd   |1010111| OP-V (OPIVV)
  funct6   | vm  |   vs2    |    vs1    | 0 0 1 |    vd   |1010111| OP-V (OPFVV)
  funct6   | vm  |   vs2    |    vs1    | 0 1 0 |  vd/rd  |1010111| OP-V (OPMVV)
  funct6   | vm  |   vs2    |   simm5   | 0 1 1 |    vd   |1010111| OP-V (OPIVI)
  funct6   | vm  |   vs2    |    rs1    | 1 0 0 |    vd   |1010111| OP-V (OPIVX)
  funct6   | vm  |   vs2    |    rs1    | 1 0 1 |    vd   |1010111| OP-V (OPFVF)
  funct6   | vm  |   vs2    |    rs1    | 1 1 0 |  vd/rd  |1010111| OP-V (OPMVX)
     6        1       5          5         3        5       7
```

```
Formats for Vector Configuration Instructions under OP-V major opcode

 31 30         25 24      20 19      15 14   12 11      7 6     0
 0 |        zimm[10:0]       |    rs1    | 1 1 1 |    rd   |1010111| vsetvli
 1 |   000000    |   rs2     |    rs1    | 1 1 1 |    rd   |1010111| vsetvl
 1        6           5          5         3        5       7
```

1 背景介绍

RISC-V向量扩展编译器支持现状——GCC



目前GCC上已经完成了对RISC-V向量扩展的当前稳定版本v0.7.1的支持，且在持续更新至最新的v0.8版本。

1 背景介绍

RISC-V向量扩展编译器支持现状——LLVM

- 目前LLVM上已经完成了对RISC-V的IMAFDC这些模块的支持，但是暂时还没有向量扩展V的支持。
- Robin Kruppe先前初步完成了一个仅包含几条向量扩展指令的基于旧版本的实现，为我们的工作提供了一些参考。

CONTENT

目录

2 向量扩展的汇编实现

TableGen

- RISC-V向量扩展的汇编支持主要是通过描述后端的寄存器信息与指令信息来实现的。其中大部分使用TableGen完成，一小部分需要C++完成。
- LLVM使用TableGen来描述领域特定的信息记录，包括编译器后端的很多特征。
- TableGen的语法类似于C++的template，用classes和definitions描述后端特征。

2 向量扩展的汇编实现

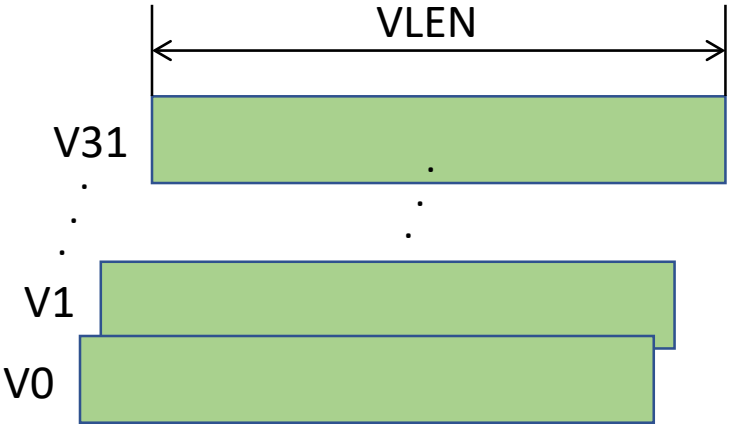
汇编实现概览

- **描述寄存器信息**
  1. 32个向量寄存器
  2. 5个控制与状态寄存器
  3. 向量掩码寄存器

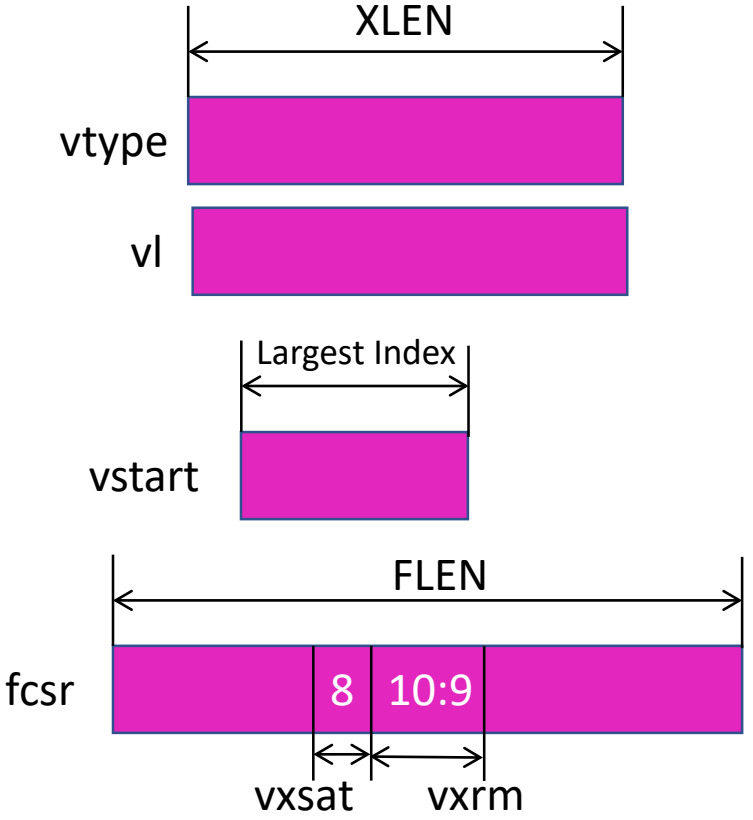
- **描述指令信息**
  1. 特殊操作数
  2. 定义指令形式（Format）
  3. 定义指令模板（Template）
  4. 定义指令（Instruction）

2 向量扩展的汇编实现

RISC-V向量扩展寄存器



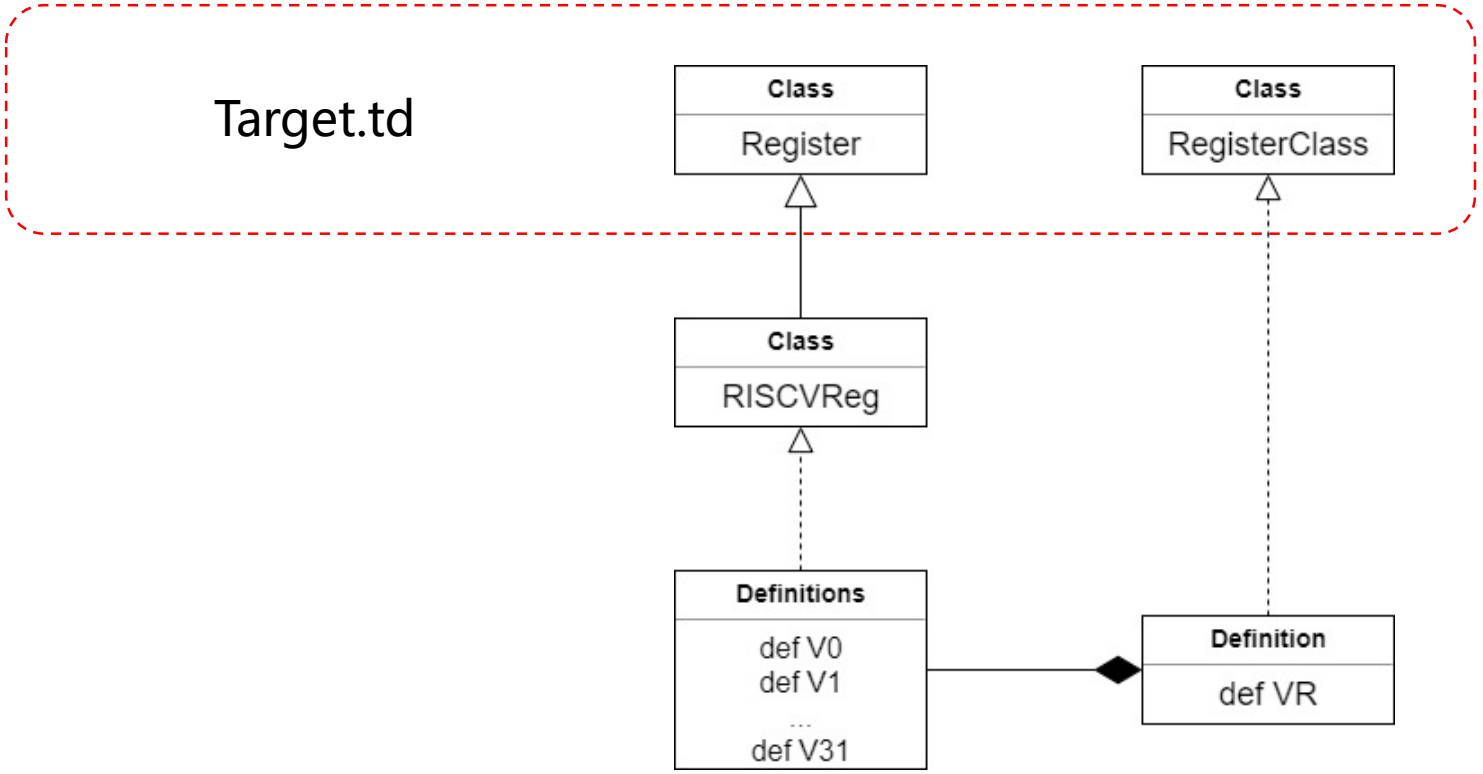
32个向量寄存器



5个控制与状态寄存器

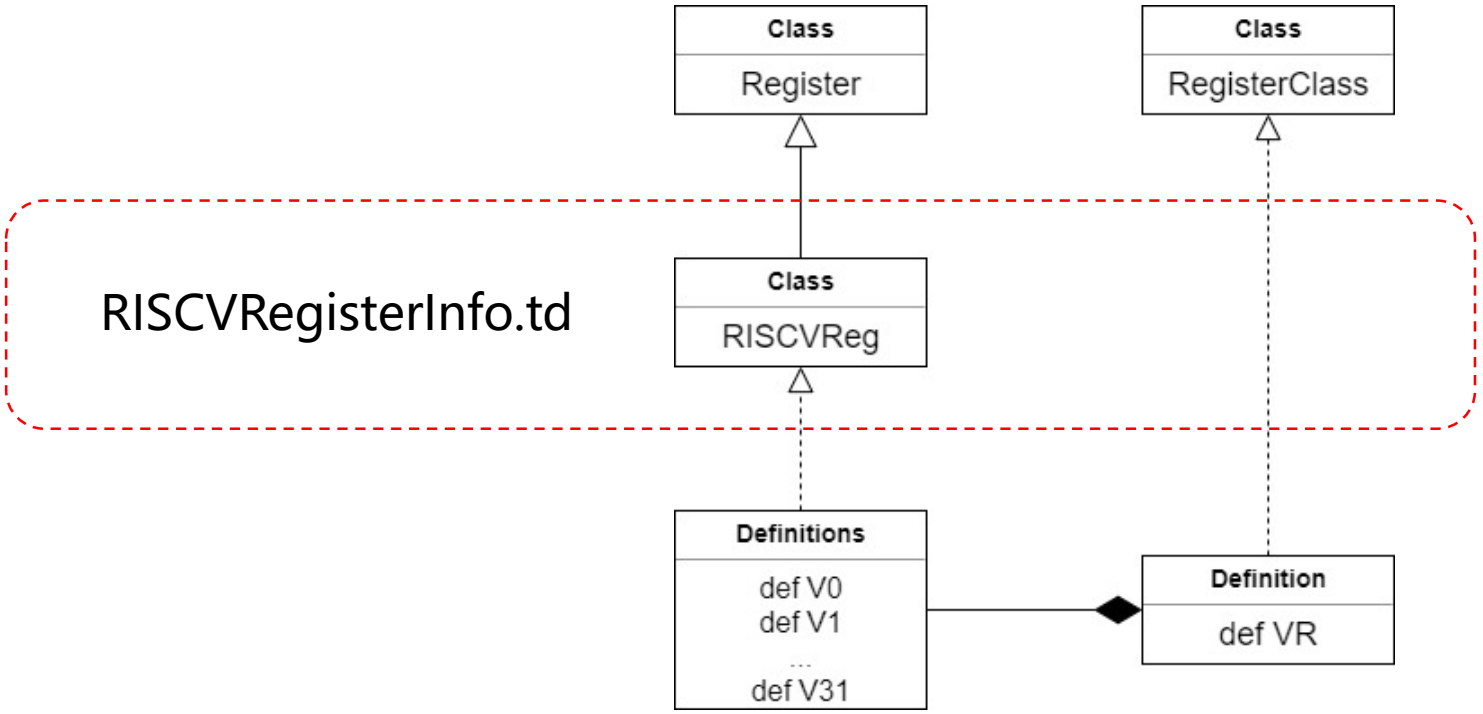
2 向量扩展的汇编实现

描述寄存器信息



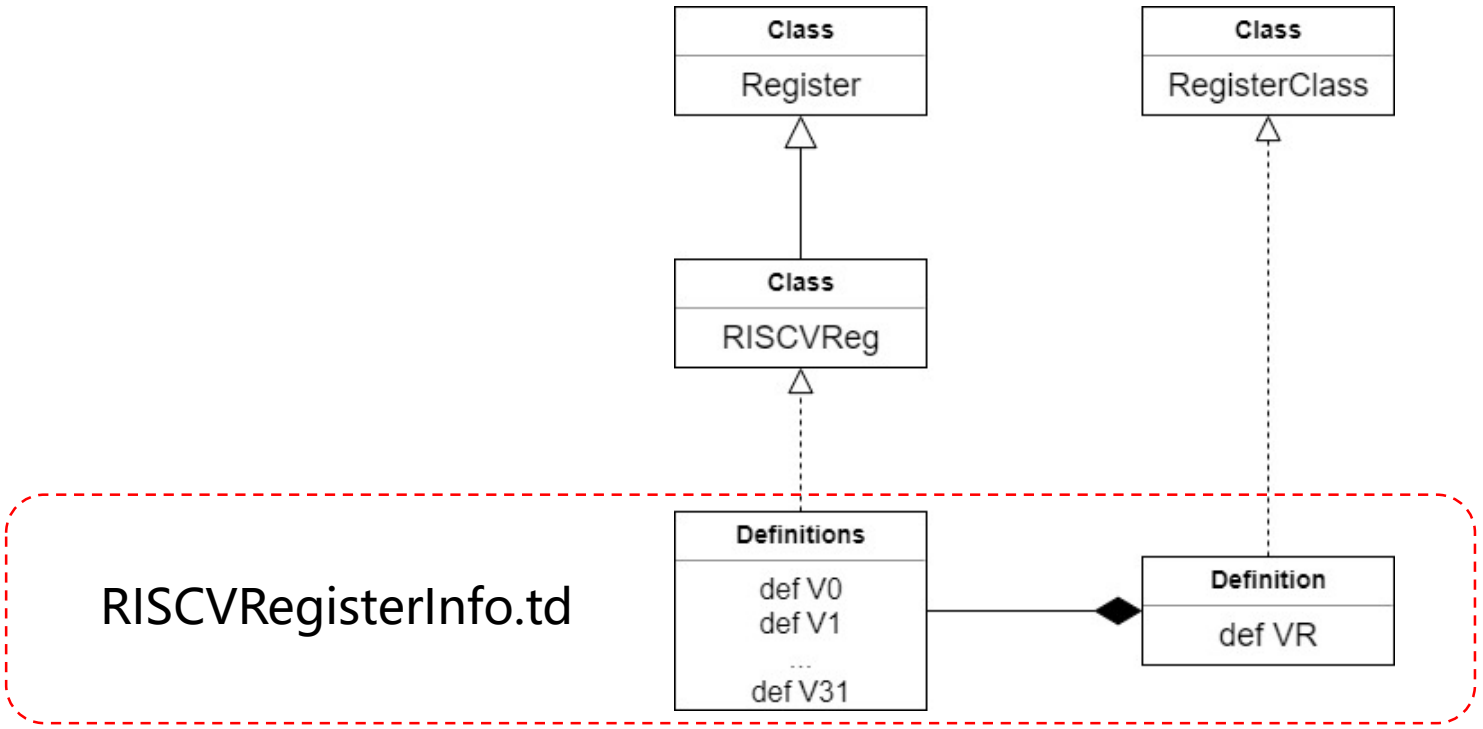
2 向量扩展的汇编实现

描述寄存器信息



2 向量扩展的汇编实现

描述寄存器信息



2 向量扩展的汇编实现

描述寄存器信息——An example

Register

RISCVRegisterInfo.td:

```
// Vector registers
let RegAltNameIndices = [ABIRegAltName] in {
  def V0  : RISCVReg<0, "v0", ["v0"]>;
  def V1  : RISCVReg<1, "v1", ["v1"]>;
  ...
  def V31 : RISCVReg<31,"v31", ["v31"]>;
}

// Vector register class
def VR : RegisterClass<"RISCV", [nxv1i32], 32, (sequence "V%u", 0, 31)>;
```

2 向量扩展的汇编实现

描述寄存器信息——An example

RISCVRegisterInfo.td:

```
// Vector registers
let RegAltNameIndices = [ABIRegAltName] in {
  def V0  : RISCVReg<0, "v0", ["v0"]>;
  def V1  : RISCVReg<1, "v1", ["v1"]>;
  ...
  def V31 : RISCVReg<31,"v31", ["v31"]>;
}
```

RegisterClass

```
// Vector register class
def VR : RegisterClass<"RISCV", [nxv1i32], 32, (sequence "V%u", 0, 31)>;
```

2 向量扩展的汇编实现

RISC-V向量扩展指令

共662条向量指令

向量指令形式 ⟶

```
Format for Vector Load Instructions under LOAD-FP major opcode
31 29 28 26  25  24      20 19       15 14   12 11      7 6     0
 nf   | mop | vm |  lumop   |    rs1    | width |    vd   |0000111| VL*  unit-stride
 nf   | mop | vm |   rs2    |    rs1    | width |    vd   |0000111| VLS* strided
 nf   | mop | vm |   vs2    |    rs1    | width |    vd   |0000111| VLX* indexed
  3     3     1      5          5         3        5       7

Format for Vector Store Instructions under STORE-FP major opcode
31 29 28 26  25  24      20 19       15 14   12 11      7 6     0
 nf   | mop | vm |  sumop   |    rs1    | width |   vs3   |0100111| VS*  unit-stride
 nf   | mop | vm |   rs2    |    rs1    | width |   vs3   |0100111| VSS* strided
 nf   | mop | vm |   vs2    |    rs1    | width |   vs3   |0100111| VSX* indexed
  3     3     1      5          5         3        5       7
```

```
Format for Vector AMO Instructions under AMO major opcode
31    27 26  25  24      20 19       15 14   12 11      7 6     0
 amoop  |wd| vm |   vs2    |    rs1    | width | vs3/vd  |0101111| VAMO*
   5      1   1      5          5         3        5       7
```

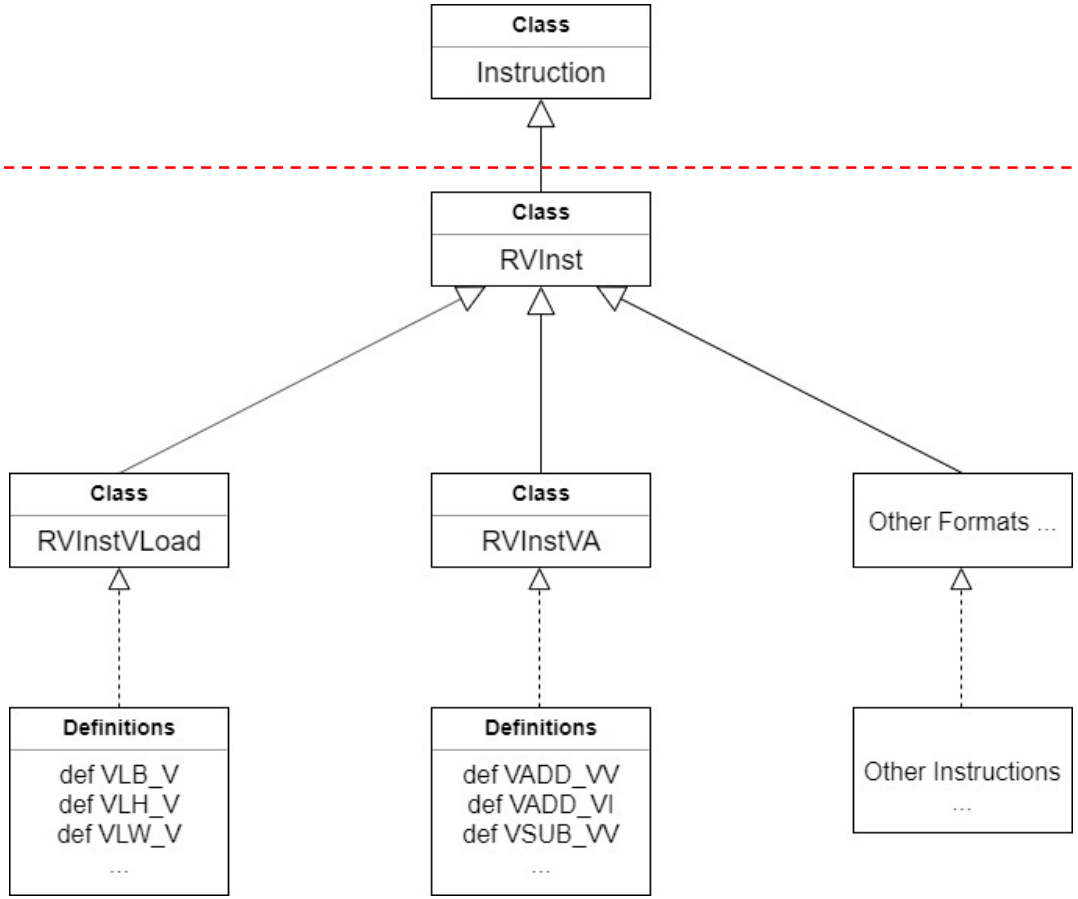
```
Formats for Vector Arithmetic Instructions under OP-V major opcode

31       26  25   24      20 19      15 14   12 11      7 6     0
  funct6   | vm  |   vs2    |    vs1    | 0 0 0 |    vd   |1010111| OP-V (OPIVV)
  funct6   | vm  |   vs2    |    vs1    | 0 0 1 |    vd   |1010111| OP-V (OPFVV)
  funct6   | vm  |   vs2    |    vs1    | 0 1 0 |  vd/rd  |1010111| OP-V (OPMVV)
  funct6   | vm  |   vs2    |   simm5   | 0 1 1 |    vd   |1010111| OP-V (OPIVI)
  funct6   | vm  |   vs2    |    rs1    | 1 0 0 |    vd   |1010111| OP-V (OPIVX)
  funct6   | vm  |   vs2    |    rs1    | 1 0 1 |    vd   |1010111| OP-V (OPFVF)
  funct6   | vm  |   vs2    |    rs1    | 1 1 0 |  vd/rd  |1010111| OP-V (OPMVX)
     6       1       5         5         3        5       7
```

```
Formats for Vector Configuration Instructions under OP-V major opcode

 31 30         25 24      20 19      15 14   12 11      7 6     0
 0 |        zimm[10:0]       |    rs1    | 1 1 1 |    rd   |1010111| vsetvli
 1 |   000000     |   rs2    |    rs1    | 1 1 1 |    rd   |1010111| vsetvl
 1       6           5         5         3        5       7
```

2 向量扩展的汇编实现

描述指令信息

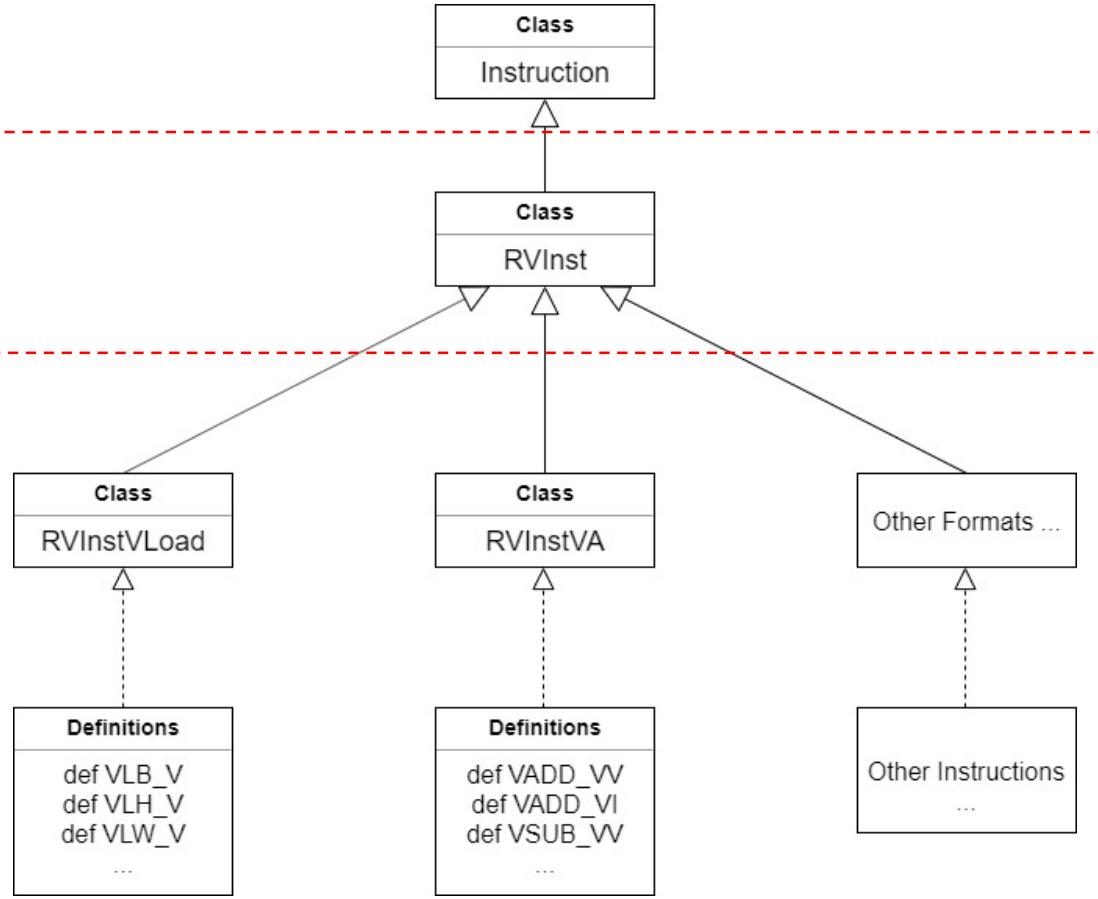
Target.td



2 向量扩展的汇编实现

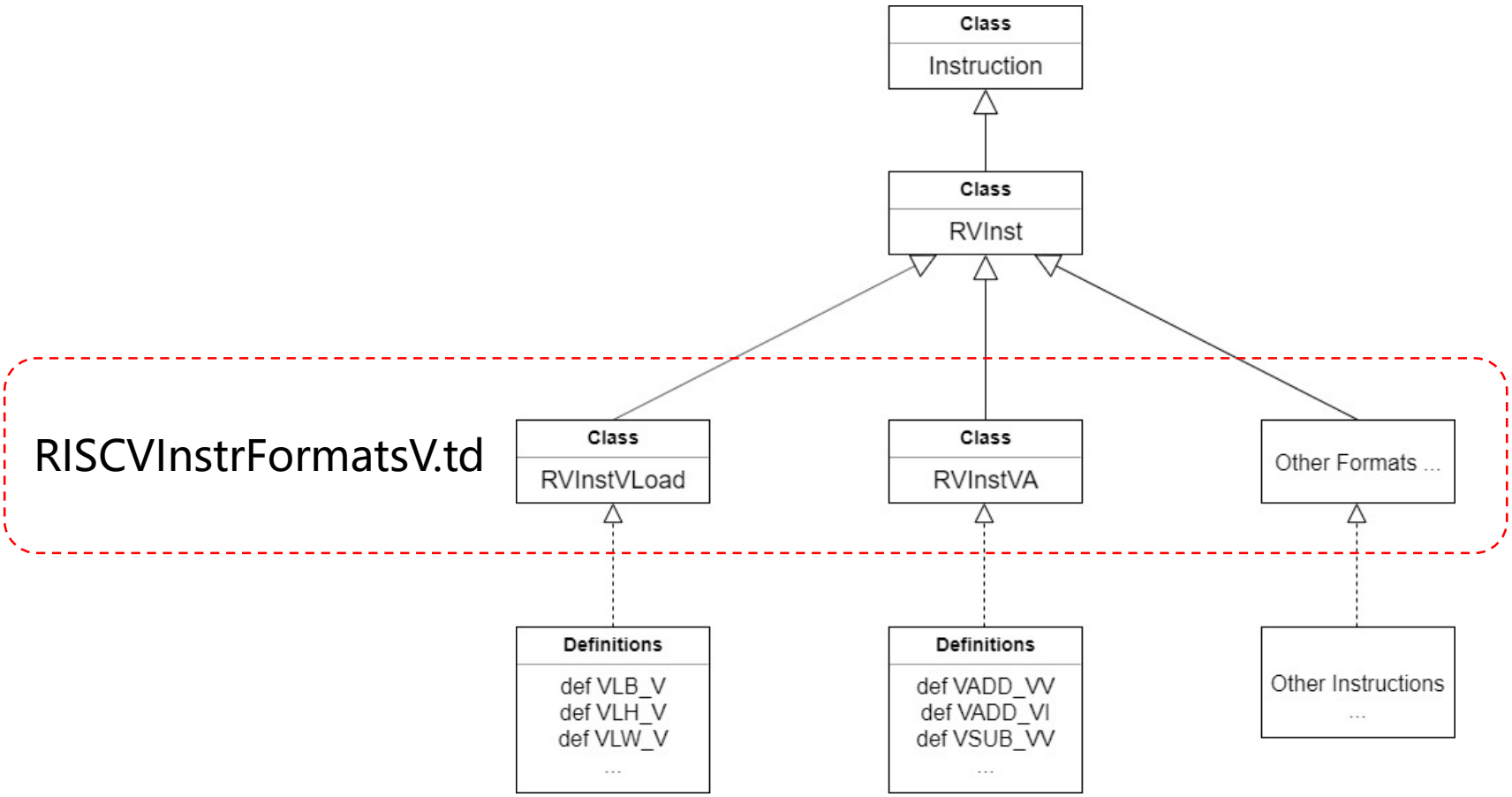
描述指令信息

RISCVInstrFormats.td



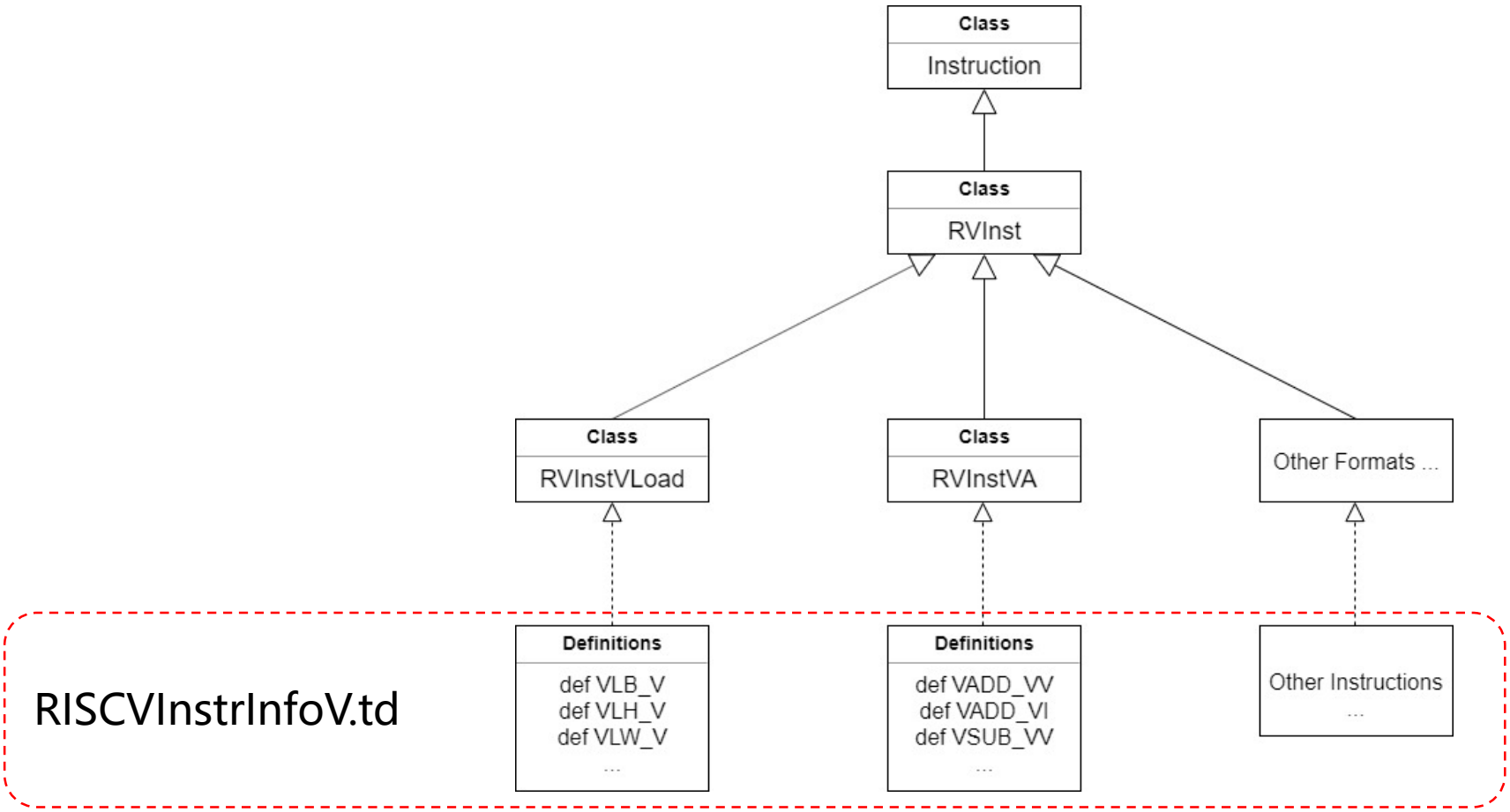
2 向量扩展的汇编实现

描述指令信息



2 向量扩展的汇编实现

描述指令信息



2 向量扩展的汇编实现

描述指令信息——An example

```
Formats for Vector Arithmetic Instructions under OP-V major opcode

31       26  25   24      20 19      15 14   12 11      7 6     0
  funct6   | vm  |   vs2    |    vs1   | 0 0 0 |    vd   |1010111| OP-V (OPIVV)
  funct6   | vm  |   vs2    |    vs1   | 0 0 1 |    vd   |1010111| OP-V (OPFVV)
  funct6   | vm  |   vs2    |    vs1   | 0 1 0 |  vd/rd  |1010111| OP-V (OPMVV)
  funct6   | vm  |   vs2    |   simm5  | 0 1 1 |    vd   |1010111| OP-V (OPIVI)
  funct6   | vm  |   vs2    |    rs1   | 1 0 0 |    vd   |1010111| OP-V (OPIVX)
  funct6   | vm  |   vs2    |    rs1   | 1 0 1 |    vd   |1010111| OP-V (OPFVF)
  funct6   | vm  |   vs2    |    rs1   | 1 1 0 |  vd/rd  |1010111| OP-V (OPMVX)
     6        1       5          5        3        5        7
```

```
Formats for Vector Configuration Instructions under OP-V major opcode

 31 30          25 24      20 19      15 14   12 11      7 6     0
 0 |        zimm[10:0]       |    rs1   | 1 1 1 |    rd   |1010111| vsetvli
 1 |   000000     |   rs2    |    rs1   | 1 1 1 |    rd   |1010111| vsetvl
 1        6            5          5        3        5        7
```

2 向量扩展的汇编实现

描述指令信息——An example

RISCVInstrFormatsV.td:

```
// Formats for Vector Arithmetic Instructions under OP-V major opcode
class RVInstVA<bits<6> funct6, bits<3> funct3, RVVMaskCond m, RISCVOpco
               dag outs, dag ins, string opcodestr, string argstr>
    : RVInst<outs, ins, opcodestr, argstr, [], InstFormatOther> {
  bits<5> vs2;

  let Inst{31-26} = funct6;
  let Inst{25}    = m.Value;
  let Inst{24-20} = vs2;
  let Inst{14-12} = funct3;
  let Opcode = opcode.Value;
}
```

2 向量扩展的汇编实现

描述指令信息——An example

```
Formats for Vector Arithmetic Instructions under OP-V major opcode

31       26  25   24      20 19      15 14   12 11      7 6     0
  funct6   | vm  |   vs2    |    vs1   | 0 0 0 |    vd   |1010111| OP-V (OPIVV)
  funct6   | vm  |   vs2    |    vs1   | 0 0 1 |    vd   |1010111| OP-V (OPFVV)
  funct6   | vm  |   vs2    |    vs1   | 0 1 0 |  vd/rd  |1010111| OP-V (OPMVV)
  funct6   | vm  |   vs2    |   simm5  | 0 1 1 |    vd   |1010111| OP-V (OPIVI)
  funct6   | vm  |   vs2    |    rs1   | 1 0 0 |    vd   |1010111| OP-V (OPIVX)
  funct6   | vm  |   vs2    |    rs1   | 1 0 1 |    vd   |1010111| OP-V (OPFVF)
  funct6   | vm  |   vs2    |    rs1   | 1 1 0 |  vd/rd  |1010111| OP-V (OPMVX)
     6        1       5          5        3        5        7
```

```
Formats for Vector Configuration Instructions under OP-V major opcode

 31 30          25 24      20 19      15 14   12 11      7 6     0
 0 |        zimm[10:0]       |    rs1   | 1 1 1 |    rd   |1010111| vsetvli
 1 |   000000     |   rs2    |    rs1   | 1 1 1 |    rd   |1010111| vsetvl
 1        6            5          5        3        5        7
```

2 向量扩展的汇编实现

描述指令信息——An example

RISCVInstrInfoV.td:

```
//===----------------------------------------------------------------------===//
// Instruction class templates
//===----------------------------------------------------------------------===//

...

let hasSideEffects = 0, mayLoad = 0, mayStore = 0 in
class VALU_OPIVV<bits<6> funct6, string opcodestr>
    : RVInstVA<funct6, 0b000, RVV_Unmasked, OPC_OP_V,
               (outs VR:$vd), (ins VR:$vs1, VR:$vs2, VLR:$vl),
               opcodestr, "$vd, $vs2, $vs1">
{
  //let Uses = [VCFG];
}
```

2 向量扩展的汇编实现

描述指令信息——An example

RISCVInstrInfoV.td:

```
let Predicates = [HasStdExtV] in {

def VADD_VV : VALU_OPIVV<0b000000, "vadd.vv">;

def VSUB_VV : VALU_OPIVV<0b000010, "vsub.vv">;

def VADD_VI : VALU_OPIVI<0b000000, "vadd.vi">;
```

vadd.vv - - - - - -向量向量加法
vsub.vv - - - - - -向量向量减法
vadd.vi - - - - - -向量立即数加法

2 向量扩展的汇编实现

描述指令信息——Mask

- 大部分的向量指令都支持掩码（Mask）操作
- TableGen提供了一个multiclass关键字，方便我们以添加后缀的形式同时定义多条记录
- 我们可以使用multiclass更方便地定义指令的掩码与非掩码形式

2 向量扩展的汇编实现

描述指令信息——Mask example

指令的掩码形式：

VADD_VV_m

指令的非掩码形式：

VADD_VV_um

RISCVInstrInfoV.td:

```
let hasSideEffects = 0, mayLoad = 0, mayStore = 0 in
multiclass VALU_OPIVVM<bits<6> funct6, string opcodestr>
{
  def _m :  RVInstVA<funct6, 0b000, RVV_Masked, OPC_OP_V,
                (outs VR:$vd), (ins VR:$vs1, VR:$vs2, VMR:$vm, VLR:$vl),
                opcodestr, "$vd, $vs2, $vs1, $vm">
  {
    bits<5> vs1;
    let Inst{19-15} = vs1;
  }
  def _um : RVInstVA<funct6, 0b000, RVV_Unmasked, OPC_OP_V,
                (outs VR:$vd), (ins VR:$vs1, VR:$vs2, VLR:$vl),
                opcodestr, "$vd, $vs2, $vs1">
  {
    bits<5> vs1;
    let Inst{19-15} = vs1;
  }
}

...

defm VADD_VV : VALU_OPIVVM<0b000000, "vadd.vv">;
```

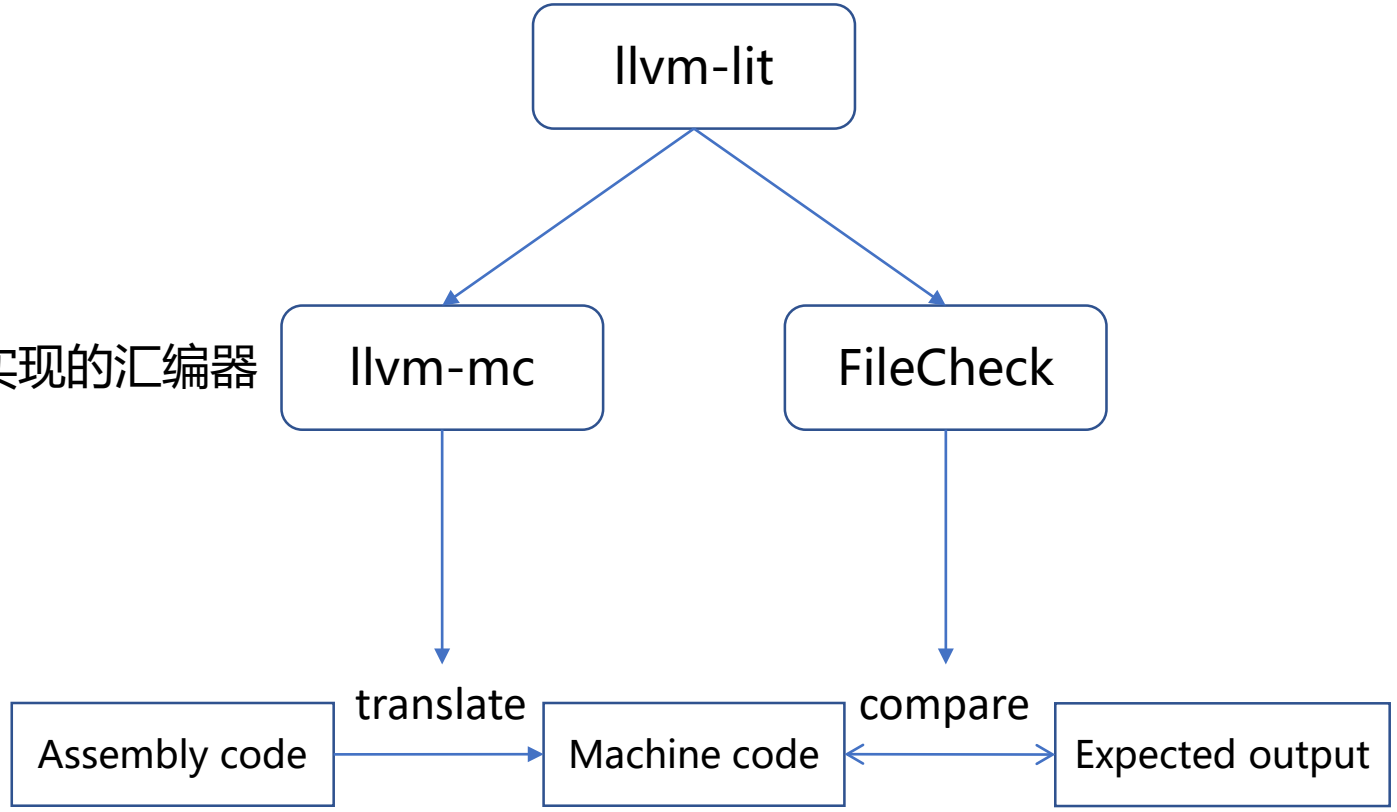
CONTENT

目录

3 汇编器测试

LLVM测试框架

我们实现的汇编器



3 汇编器测试

汇编测试用例

rvv-valid.s:

```
# CHECK-ASM-AND-OBJ: vadd.vv v0, v1, v0
# CHECK-ASM: encoding: [0x57,0x00,0x10,0x02]
vadd.vv v0, v1, v0
```

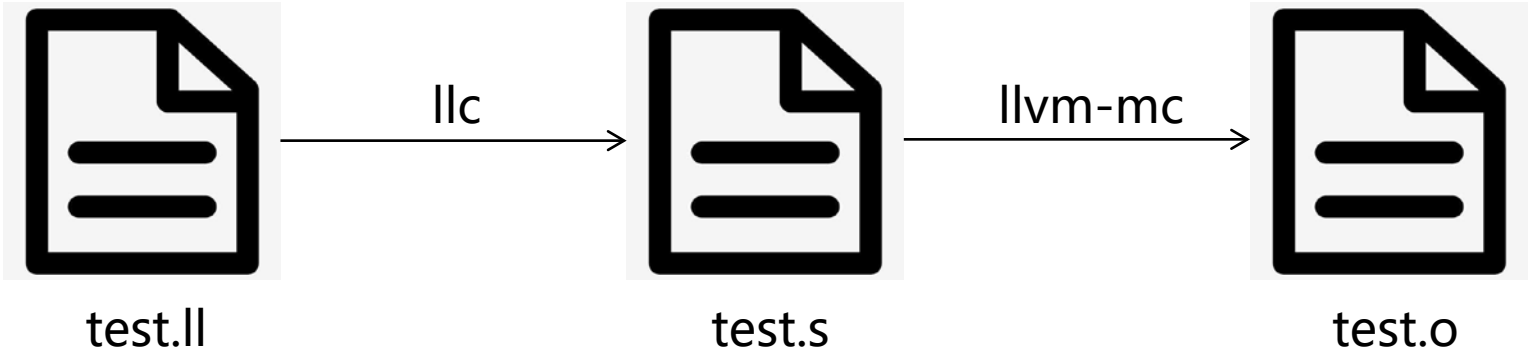
预期的结果在带有特定前缀的注释内给出。

CONTENT

目录

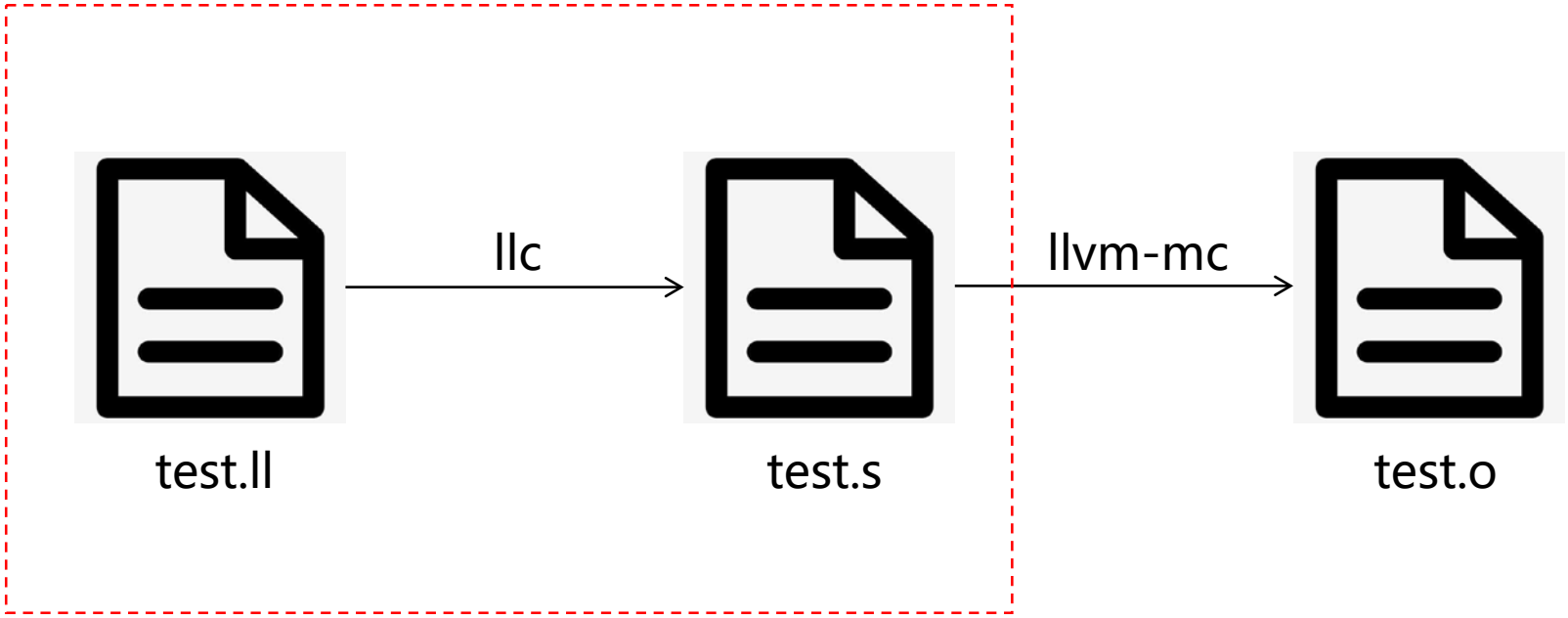
4 后续工作

Intrinsic函数



4 后续工作

Intrinsic函数



4 后续工作

Intrinsic函数

- 定义Intrinsic函数
- 使用llvm的patterns将Intrinsic函数与指令配对

IntrinsicsRISCV.td:

```
def int_riscv_vadd : Intrinsic<[llvm_nxv1i32_ty],
                               [llvm_nxv1i32_ty, llvm_nxv1i32_ty, llvm_i32_ty],
                               [IntrNoMem]>;
```

4 后续工作

Intrinsic函数

- 定义Intrinsic函数
- 使用llvm的patterns将Intrinsic函数与指令配对

RISCVInstrInfoV.td:

```
def : PatVrVr<int_riscv_vadd, VADD_VV>;
```

4 后续工作

自动向量化

- 自动向量化是在LLVM-IR上的优化工作
- 目前已有的LLVM Pass针对长度可变的向量扩展(SVE)进行向量化时存在一些问题
- 我们需要实现一个（或几个）额外的Pass来完成RISC-V向量扩展的自动向量化
- 后续我们会针对向量化的效果进行更多的优化

4 后续工作

保持版本更新

- RISC-V的向量扩展规范在12月14日发布了新的稳定版本v0.8
- 我们目前的实现基于版本v0.7.1
- 我们后续也会持续跟踪版本更新情况保证代码兼容最新版本

谢 谢