独自ベクトル処理機能を備えたプロセッサ向け 自動ベクトル化コンパイラの開発

永池 晃太朗 † 大津 金光 †† 横田 隆史 ††
† 宇都宮大学工学部情報工学科 †† 宇都宮大学大学院地域創生科学研究科

1 はじめに

我々は機械語コードの変更なしに同時並列演算数を変更することができる，データ並列処理のためのスケーラブルなベクトル拡張を実現したベクトル拡張付き RISC-V を提案している [1].

我々のベクトル拡張付き RISC-V は，ARM のベクトル拡張である ARM SVE[2] を参考に組込み向けに RISC-V[3] をベクトル拡張したものである．これにより，機械語コードが同時演算数に依存しないスケーラブルなベクトル拡張を実現したが，ベクトル拡張に対応したコンパイラがない．アセンブリコードを生成は ARM SVE のコンパイラによる AR SVE アセンブリコードを変換するといった手法を取っており，規模の大きいプログラム等の場合，アセンブリコードを得ることは容易ではない.

この問題に対して，解決策として既存の RISC-V コンパイラに変更を加えることによって，ベクトル拡張付き RISC-V アセンブリコードを得ることのできるコンパイラを開発するという手段を考えた.

コンパイラの開発はコンパイラ基盤である LLVM[4] を用いて行う．コンパイラ基盤はコンパイラの機能がモジュール化されており，既存機能の再利用が可能になっている．そのため，コンパイラの開発の際にコンパイラ基盤を用いることによって独自部分のみの開発で済む．前述した通りベクトル拡張付き RISC-V は RISC-V を拡張したものであることから，LLVM の RISC-V コンパイラとしての機能を再利用してコンパイラの開発を行う.

なお，使用する LLVM のバージョンは 13.0.0 である.

2 ベクトル拡張付き RISC-V

ベクトル拡張付き RISC-V は RISC-V が有する 4 つのカスタム命令のためのオペコード領域のうち 2 つを利用しており，1 つをベクトルロード，ストア命令，もう一方をベクトル演算命令とベクトル制御命令に使用している．ベクトルロード，ストア命令のアドレスはベースとなるスカラレジスタの値にオフセットを加えることで計算する．オフセットはスカラレジスタ，ベクトルレジスタ，即値の 3 種類を指定できる．オフセットにベクトルレジスタを指定することでギャザー，スキャターによる非連続なロード，ストアを行う.

ベクトル演算命令は，プレディケートあり演算，プレディケートなし演算命令，即値による演算命令に分けられる．演算命令は，基本的な算術論理演算命令，乗除算命令，ベクトルレジスタの各要素の総和を求める命令やアドレス計算の際に用いるインデックスレジスタを生成する命令からなる．プレディケートあり演算命令ではプレディケートレジスタによるベクトルマスク制御を行う.

ベクトル制御命令はプレディケート演算命令とベクトル長操作命令に分けられる．プレディケート演算命令は，プレディケートレジスタ同士の論理演算を行う．ベクトル長操作命令は，スカラレジスタに対しベクトル長を足す等の命令がある.

以上の命令は ARM SVE を参考にした命令であるが，その命令フォーマットは RISC-V の命令形式に従ったものである.

レジスタの構成は，v0 から v31 のベクトルレジスタ，vp0 から vp15 のプレディケートレジスタ，ベクトル長レジスタ，x0 から x31 の汎用レジスタとなっている.

また，ベクトル拡張付き RISC-V の基本命令は RV32I の命令を組み込んでいる.

3 LLVM コンパイラ基盤

LLVM の構成を図に示す．LLVM では C 言語などのソースコードから中間表現である LLVM IR に変換を行うフロントエンド，LLVM IR に対して最適化，アセンブリコード生成を行うバックエンドからなる．本研究では独自命令生成のためにバックエンドに対して変更を加える.

バックエンドにおけるコード生成について，処理の流れと実行されるパスを図に示す．LLVM バックエンドでは LLVM IR から形式を DAG 形式，MachineCode 形式へと変化させながらそれぞれの処理を行う．SelectionDAG では LLVM IR を DAG 形式へと変換し，DAG のパターンマッチングによる命令の単純化，共通部分削除などの最適化，ターゲット命令への変換の順に行う．SelectionDAG は命令変換を行った後に MachineCode 形式を出力する．MachineCode 形式はターゲットが使用する実際の命令を持った形式であり，SSA ベースの最適化を行い，命令への物理レジスタの割当を行う．レジスタの割当を行った後は非 SSA 形式となる．MC Layer はアセンブリコード，オブジェクトコードを出力するための命令形式であり，関数などの構造がない.

以上の LLVM バックエンドでの処理はレジスタ情報，命令フォーマットや命令などのターゲット情報に従って行われる．これらのターゲット情報は LLVM に

Development of automatic vectorization compiler for processors with original vector processing function.
†Nagaike Kotaro, ††Ootsu Kanemitsu, ††Yokota Takashi, Department of Information Science, Faculty of Engineering, Utsunomiya University (†)

おけるターゲット情報記述のためのドメイン固有言語である TableGen によって行われる．

LLVM では RISC-V の V 拡張用に TableGen によるベクトルレジスタ，ベクトル命令の定義が既に存在している．本研究ではベクトルレジスタはそのまま使用し，ベクトル命令については既存の定義に従った形式でベクトル拡張付き RISC-V 命令の定義を行う．

また，ベクトル命令生成のための機能として LLVM には自動ベクトル化機能が備わっている．フロントエンドにて入力ソースコードの繰り返しによる処理をベクトル化された LLVM IR に変換を行う．ベクトル化された LLVM コードからパターンマッチングにより各ターゲットへのベクトル命令への変換などが行われる．RISC-V の V 拡張用のパターンマッチングについても実装済みのものを再利用することでベクトル拡張付き RISC-V 命令の生成する．

4 LLVM バックエンドにおける独自命令の生成機能の実装

謝辞

本研究は，一部日本学術振興会科学研究費補助金 (基盤研究 (C)24500055，同 (C)15K00068) の援助による．

参考文献

[1] Yoshiki Kimura,Tomoya Kikuchi,Kanemitsu Ootsu,Takashi Yokota: “Proposal of Scalable Vector Instruc-tion Set for Embedded RISC-V Processor,”, Proc. 2019 Seventh International Symposium on Computing and Networking Workshops (CANDARW), Vol.1, pp.435-439, 2019.

[2] Nigel Stephens, et al: “The ARM Scalable Vector Extension,”, IEEE Micro, Vol.37, No.2, pp.26-39, 2017.