5H-05

ARM SVEからベクトル拡張付きRISC-Vへのアセンブリコードトランスレータの検討

土屋 達哉 † 木村 嘉毅 †† 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が提案している[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(Scalable Vector Extension)[2]の命令セットを参考に組込み向けにRISC-V[3]に拡張したものである．これにより，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．機械語コードを得る手段として，RISC-Vのコンパイラをベクトル拡張に対応したものに改良するという方法が考えられるが，ベクトル化コンパイラの開発に時間がかかり，難易度も高い．

この問題に対して，ARM SVE向けベクトル化コンパイラが生成するアセンブリソースプログラムを利用して，我々のシステム向けのベクトル化アセンブリコードを得る方法であれば短期間に開発できると考えた．そのアセンブリコードをRISC-Vアセンブラでアセンブルし RISC-Vベクトル化機械語コードを生成する．以上のようにすることでRISC-Vベクトル化機械語コードを生成できる．本稿ではアセンブリコードトランスレータにおける命令変換方法について検討する．

2 命令セットアーキテクチャ(ISA)

ARM SVEとベクトル拡張付きRISC-Vでは，備わっているレジスタが違ったり，レジスタ利用規約が違う等の考慮すべき問題がある．そのため，ARM SVEとベクトル拡張付きRISC-Vのレジスタについて確認する．ARM SVEハイパフォーマンス用途を想定したARMv8-A AArch64(64bit）命令セットのHPC向けベクトル拡張で，SVEの特徴としては，ベクトル長を固定しないという点がある．そのレジスタ構成はZ0からZ31までの32本のベクトルレジスタ，P0からP15までの16本のプレディケートレジスタ，AArch64に備わっていたx0からx30までの汎用レジスタ，浮動小数点レジスタ，FPCRレジスタ，FPSRレジスタ，スタックポインタ，プログラムカウンタ，フラグレジスタがある．ベクトル長は128bitから2048bitまで，設定できる．汎用レジスタの幅は64bitであり64bitレジスタとして使う場合はx1，x2，32bitレジスタとして使う場合はw1，w2というように表示する．汎用レジスタとベクトルレジスタについてレジスタ利用規約がある．

我々のベクトル拡張付きRISC-VはARM SVEを参考に組込み向けにベクトル拡張したもので，そのレジスタ構成はv0からv31までの32本のベクトルレジスタ，vp0からvp15まで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までの32本の汎用レジスタ，プログラムカウンタがある．組み込み向けを想定しているためSVEの一部の命令を採用している．ベクトルレジスタの長さは128×2^nbitであり，汎用レジスタの幅は32bitとなっている．ARM SVEの64bit汎用レジスタを扱う命令を変換する際は，32bitの範囲を使用することを前提として上位32bitを無視して扱う．RISC-Vの汎用レジスタについてもレジスタ利用規約がある．ベクトルレジスタに関してはレジスタの規約がない．ベクトル拡張付きRISC-Vは基本命令に関してはRV32Iのみ組み込んでいるため，RISC-Vの命令に変換する際にはRV32Iの命令のみを使う．

3 アセンブリ命令の対応関係

C言語で記述されたプログラムをARM SVEコンパイラであるGCC(GNU Compiler Collection)-10.2.0[4]を用いてベクトル化を行い，ARM SVEのベクトル化アセンブリコードを生成した．生成されたARM SVEのアセンブリコードがアセンブリコードトランスレータの入力ファイルとなる．アセンブリコードトランスレータではARM SVEの命令を一行ずつARM SVEとベクトル拡張付き RISC-Vの命令対応表を元に変換する．そうしてできたRISC-Vのベクトル化アセンブリコードがアセンブリコードトランスレータの出力ファイルとなる．

アセンブリコードトランスレータによる変換の例として図1に配列間の加算を行うプログラムのARM SVEとベクトル拡張付きRISC-Vのアセンブリコードを示す．ARM SVEのこの例ではx3レジスタがループカウンタ，w4レジスタが配列長を格納するレジスタとして使われている．最初の行ではループカウンタx3が初期化され，2行目では配列長999がw4に設定されている．3行目のwhilelo命令でプレディケートレジスタが初期化されている．4，5行目で配列の各要素をメモリからベクトルレジスタに読み込み，6行目で加算を行う．7行目の命令で結果をメモリに格納する．8行目でループカウンタをベクトル要素数だけインクリメントし、9行目でループカウンタと配列長を

Examination of assembly code translator from ARM SVE to RISC-V with vector extension
†Tatsuya Tsuchiya, ††Yoshiki Kimura, ††Kanemitsu Ootsu, ††Takashi Yokota,
Department of Information Science, Faculty of Engineering, Utsunomiya University (†)
Graduate School of Regional Development and Creativity, Utsunomiya University (††)

	ARM SVE			ベクトル拡張付きRISC-V		
1	ADD:	mov	x3, 0	ADD:	li	a3, 0
2		mov	w4, 999		li	a4, 999
3		whilelo	p0.s, wzr, w4		vptrue.w	vp0
4	LOOP:	ld1w	z0.s, p0/z, [x0, x3, lsl 2]	LOOP:	vlw.w	v0, vp0, a0, a3
5		ld1w	z1.s, p0/z, [x1, x3, lsl 2]		vlw.w	v1, vp0, a1, a3
6		add	z0.s, z0.s, z1.s		vadd.w	v0, vp0, v1
7		st1w	z0.s, p0, [x2, x3, lsl 2]		vsw.w	v0, vp0, a2, a3
8		incw	x3		vincw	a3
9		whilelo	p0.s, w3, w4		vwhilelo.w	vp0, a3, a4
10		b.any	LOOP		blt	a3, a4, LOOP

図 1: ARM SVE からベクトル拡張付き RISC-V へのアセンブリコード変換例

比較し，プレディケートレジスタを設定している．最後の行ではループの条件判定を行っている．

命令ニーモニックで v がつく命令がベクトル拡張命令であり，命令の後ろに.<size>をつけることでベクトル要素のサイズを指定する．ARM SVE で使用している汎用レジスタは x0，x1，x2，x3，x4 レジスタで，レジスタの規約では callee save レジスタとなっている．ベクトル拡張付き RISC-V では同じ callee save レジスタである a0，a1，a2，a3，a4 に対応させる．ベクトルレジスタ，プレディケートレジスタについては ARM SVE で指定された番号と同じレジスタ番号のレジスタをベクトル拡張付き RISC-V で使う．この例では a3 レジスタがループカウンタ，a4 レジスタが配列長を格納するレジスタとして使う．この配列間の加算を行うプログラムの場合，各行の命令が ARM SVE の命令と基本的に 1 対 1 対応して変換する．しかし，ベクトル拡張付き RISC-V はフラグレジスタが存在していないため最後の行の ARM SVE のフラグを参照する命令を別の形に置き換える必要がある．ARM SVE の 10 行目の b.any は 9 行目の whilelo で設定されたフラグ Z の値で分岐を判断する命令である．プログラムの内容としては，ループカウンタ w3 が配列長 w4 より小さい間はループするというものである．そのため図 1 のように，b.any 命令をベクトル拡張付き RISC-V の方ではループカウンタ a3 が配列長 a4 より小さい場合指定したアドレスに飛ぶ blt 命令に置き換える．この様に ARM SVE のフラグレジスタを使った分岐命令はフラグをセットする命令のレジスタを参照する分岐命令に変換する．

4 複数命令を用いて変換する場合

異なる命令セット間での変換であるためベクトル拡張付き RISC-V は ARM SVE の全ての命令が 1 命令で置き換えられない場合もある．その場合は，ARM SVE の 1 命令を複数の RISC-V の命令を使って同等の処理に置き換えることで対応する．sub(即値) 命令は RV32I にはない．表 1 に例として sub(即値) 命令を RISC-V の複数の命令で変換した例を示す．sub(即値) 命令は rs1 で指定したレジスタの内容を imm で減算した結果を rd に入れる命令である．RISC-V 側では即値 imm を一度レジスタに入れ，レジスタ間の減算 sub を行うことで同じ動作を実現させる．

表 1: sub(即値) の変換

ARM	RISC-V
sub rd, rs1, imm	li rd, r0, imm sub rd, rs1, rd

5 おわりに

本稿では，ARM SVE アセンブリ命令から我々独自のベクトル拡張付き RISC-V アセンブリ命令への変換について検討した．今後はアセンブリコードトランスレータを実装し，生成されたアセンブリコードが正しく実行できることを確認する．

謝辞

本研究は一部 JSPS 科研費 20K11726 の援助による．

参考文献

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

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

[3] Andrew Waterman, Krste Asanović: “The RISC-V Instruction Set Manual Volume I: Unprivileged ISA”, December 13, 2019

[4] “GCC 10 Release Series Changes, New Features, and Fixes” https://gcc.gnu.org/gcc-10/changes.html 閲覧日 2021/1/8