

## FLENG の中間言語とエミュレータの実装

## 3 G-3

下山 健, 林 国輝, 小池 汎平, 田中英彦

{ken,lim,koike,tanaka}.mtl.t.u-tokyo.ac.jp

(東京大学 工学部)

## 1 はじめに

我々は現在、並列推論エンジン PIE64[4]の開発を進めている。PIE64 上で大規模な知識処理を行なうための言語として、Committed-Choice 型言語 FLENG[3]と、オブジェクト指向の概念を採り入れた上位言語 FLENG++ を採用している。FLENG には、処理系に効率良い並列処理やメモリ管理を行ない記述力、可読性を高めるために様々なアノテーションを追加した、低水準記述言語 FLENG--[7]がある。本稿で概説する中間言語は FLENG-- を PIE64 の推論プロセッサである、UNIREDI II [5] のネイティブコードに落とすための中間言語である。この中間言語は、様々な最適化が行なえるように設計されており、エミュレータによってこれらを検査できるようにになっている。最終的には UNIREDI II や汎用プロセッサのネイティブコードに落とされ高速実行される。

## 2 FLENG コンパイラと中間言語

図1に示されるように FLENG コンパイラは、FLENG を FLENG の中間言語にコンパイルする。この種類の言語としては、PROLOG では WAM[1] コード、Committed-Choice 型言語 GHC には、KL1B[2]がある。これらのコードと比較するとこの FLENG code には、以下のような特徴がある。

- 最終的に PIE64 の推論プロセッサ UNIREDI II のネイティブコードに落とすことを目的としているので、UNIREDI II の命令アーキテクチャによく整合する命令セットとなっている。
- SRA、マクロ構文等を利用した最適化 [7] がこの中間言語上で行なえるように命令の抽象化が比較的低下レベルな中間言語となっている。
- 中間言語上の命令の統合などの中間言語上の実行の高速化を主眼とせず、言語の本質的な最適化を行なうことを目的としている。

## 3 FLENG の中間言語仕様

## 3.1 ヘッドで生成される命令

ヘッドのコンパイルで生成される主な命令を、表1に示す。ヘッドで生成される命令はデリファレンス命令、タグチェック命令、値チェック命令に分けることができる。これらの命令はそれぞれ、deref 命令、checkTAG 命令、checkCONST 命令であり、後の2命令は、デリファレンス命令と合成して、derefTAG 命令、derefCONST 命令となる。デリファレンスを

<sup>0</sup>“The specifications of FLENG code and the implementation of emulator” Takeshi SHIMUYAMA, Lim Kok Hwee, Hanpei KOIKE, Hidehiko TANAKA  
the University of Tokyo

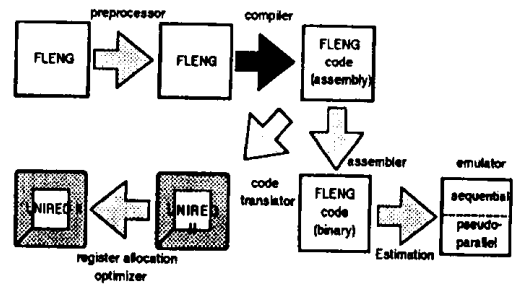


図1: FLENG コンパイラ的位置付け

行なう命令は、UNIREDI II の命令体系と適合するように、UNDEF の場合と、タグチェックで失敗した場合でそれぞれに独立に飛び先のオフセットを持てるようにした。ヘッドのユニフィケーションが決定的に行なわれる場合のサスペンドはレジスタを使って高速化できることはすでに示した。[6]一方、ヘッドのユニフィケーションが非決定的に行なわれる場合は、サスペンドの原因となり得る変数の情報をスタックにプッシュすることになる。これらの命令は、ニーモニックに "X" をつけた、derefX 系命令で行なわれ、ヘッドのユニフィケーションが非決定的に行なわれる時に生成される。表1に示した命令は、ソースオペランドとしてレジスタが指定されるが他に、レジスタとディスプレイメントでメモリを指定するアドレッシングモードもあり、これらは構造体の内部を操作するのに使われる。

## 3.2 ボディで生成される命令

ボディのコンパイルで生成される主な命令を、表2に示す。bind 命令は、指定された変数が未束縛のときにロックをかけて(並列環境時)値を書き込む動作を行なう。未束縛でないときは、offset で指定される番地にジャンプする。loadStore 命令は、メモリアドレッシングモード、レジスタライト、メモリアドレッシングモードを1

deref(X)	%s, %d, UNDEF
derefTAG(X)	%s, %d, FAIL, UNDEF
derefCONST(X)	%s, %d, data, FAIL, UNDEF
checkTAG	%s, FAIL
checkCONST	%s, data, FAIL

表1: ヘッドユニフィケーションで生成される命令

bind	%s, %d, offset
execute(X)	offset
proceed(X)	
suspend(X)	
allocHeap[1,2]	size, tag, %d
enqueueGoal	mode
(t)load	[%s+offset], %d
loadI	data, %d
store	%s, [%b+offset]
storeUNDEF	[%b+offset]
loadStore	[%s+offsetS], %d, [%b+offsetB]
branchXX	offset
arithmetic	%s, %d, %r

表 2: ポディリダクションで生成される命令 (抜粋)

命令で行なう命令であり、ヘッドユニフィケーションで参照されない変数等のポディへの転送に使用される。この命令を分解すると最終的に UNIRED II のネイティブコードで生成するのが困難になるので、中間言語上で生成するようにする。execute 命令、proceed 命令、suspend 命令の 3 命令には、ヘッドユニフィケーションが非決定的に行なわれる時に使われるスタックを初期値に戻す処理を行なう executeX 命令、proceedX 命令、suspendX 命令もそれぞれ用意されている。allocHeap 命令は、ヒープからメモリをアロケートし、指定されたタグをつける命令である。UNIRED II は、ページング GC に対応できるようにヒープが 2 つ用意されているので、この中間言語でも対応できるようにしている。enqueueGoal 命令では、並列実行時にゴールの飛び先を指定できるようにしている。これらは、並列エミュレータで実行され、負荷分散の最適化の評価を行なえるようになっている。コンパイル例を図 2 に示す。

#### 4 FLENG code からネイティブコードへの変換

FLENG コードは、その性格上抽象度が比較的 low、ネイティブコードに近い命令セットとなっている。従ってこのコードから、UNIRED II や SPARC 等の汎用プロセッサに落とすことは比較的容易であるが以下のような点が重要である。

```
append([H|T], X, Y) :- Y = [H|Z], append(T, X, Z).
```

append/3: derefLST	%0, %3, fail, suspend
allocHeap	2, LST, %4
loadStore	[%3-LST], %5, [%4-LST]
storeUNDEF	[%4-LST+4]
bind	%4, %2, generalUnify
tload	[%3-LST+4], %0
addI	%4, -LST+4+VAR, %2
execute	append/3
suspend: suspend	append/3
fail: proceed	

図 2: append/3 のコンパイル例

- アドレッシングモードが違う命令の変換。特に、UNIRED II では、deref 系命令のオフセットの指定方法等。
- 中間言語上では、レジスタを無制限に使用するので、実際のネイティブコードに落ち時は、レジスタ割り当ての最適化を行なわないと spill 処理が多く発生して非常に遅いコードになる場合が多くなると予想される。
- バイブライニングハザードを避けるように命令を並べ替える最適化

#### 5 エミュレータの実装

エミュレータは、FLENG コンパイラの出力した FLENG code のバイナリコードと、シンボルテーブルを読み込んでエミュレートする。このエミュレータには 2 種類のバージョンが用意されている。一方は、通常の逐次実行のエミュレータで中間言語の高速実行を主な目的としている。もう一方は、疑似並列環境でのエミュレータで SUN の Light Weight Process を使って書かれている。後者のエミュレータは、並列環境下でのプログラムの動作保証や並列環境での最適化の効果測定を主な目的としており、PIE64 が動作するまでに SRA、GC、マクロ構文、負荷分散戦略等の評価に使用する。

#### 6 おわりに

現在、コンパイラは一部 SUN-4 上で動作しており、エミュレータは、逐次動作版と Light Weight Process を使った疑似並列版が 12 月現在開発中である。完成次第、SRA、マクロ構文、負荷分散等の最適化の評価を行なっていく方針である。

なお、本研究は文部省特別推進研究 No.62065002 による。

#### 参考文献

- [1] Warren, D.H.D., "An Abstract Prolog Instruction Set", Tech. Note 309, SRI International, 1983.
- [2] Y.Kimura and T.Chikayama, "An Abstract KL1 Machine and its Instruction Set", Proc. of SLP'87
- [3] Nilsson M. and Tanaka H., "FLENG Prolog - The Language which turns supercomputers into Prolog machines" Logic Programming Conference, 1986, pp.209-216.
- [4] H.Koike and H.Tanaka, "Multi-Context Processing and Data Balancing Mechanism of the Parallel Inference Machine PIE64", Proc. of Fifth Generation Computer System, Tokyo, Japan, November 1988.
- [5] 島田, 下山, 清水, 小池, 田中, "推論プロセッサ UNIRED II の命令セット", 情報処理学会計算機アーキテクチャ研究会, 79-5, 1989 年 11 月
- [6] 下山, 島田, 小池, "FLENG コンパイラの最適化処理", 情報処理学会第 39 回大会 4Q-6, 1989 年 10 月
- [7] 下山, 小池, 田中, "Comitted-Choice 型言語 FLENG 上の最適化コンパイル", ソフトウェア科学会第六回大会, 1989, pp.37-40.