3G－3

FLENGの中間言語とエミュレータの実装

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、UNIREDII[5]のネイティブコードに落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評価できるようになっている。最終的にはUNIREDIIや汎用プロセッサのネイティブコードに落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UNIREDIIのネイティブコードに落すことを目的としているので、UNIREDIIの命令アーキテクチャによく整合する命令セットなっている。
- 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にしている。

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。デリファレンスを行なう命令は、UNIRED 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ス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。

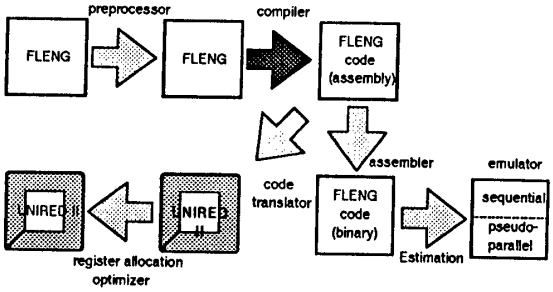

図1: FLENGコンパイラの位置付け

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、メモリリード、レジスタライト、メモリライトを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

[0] "The specifications of FLENG code and the implementation of emulator" Takeshi SHIMOYAMA, Lim Kok Hwee, Hanpei KOIKE, Hidehiko TANAKA
the University of Tokyo

```
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: ボ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ボ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UNIREDII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。UNIREDII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なっている。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なっている。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低く、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す時は、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。
- パイプライン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 Abstruct Prolog Instruction Set", Tech. Note 309, SRI International, 1983.

[2] Y.Kimura and T.Chikayama, "An Abstruct 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 Programing 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.