4J－8

機器制御用言語 el(α)コンパイラの開発

佐久間 和子[†] 渡邊 坦[†] 荒井 秀之[††] 梅谷 孝文[†††]

[†](株)日立製作所 システム開発研究所 [††]同社小田原工場 [†††]日立マイクロコンピュータエンジニアリング株式会社

1．はじめに

機器制御用言語として設計した el(α)言語[1] のコンパイラの開発を行なった。

el(α)はその使用目的に合わせて特有のアーキテクチヤを持つ種々の機器制御専用マシンを主な対象としているため、コンパイラは多機種への展開が容易でなければならない。また、制御速度に直接影響するため、アセンブリ言語にあまり劣らない性能が要求される。

el(α)には、基本的言語機能の他に、ソフトウエアの共同開発を容易にするためのライブラリ機能、オブジェクト性能の向上を図るインライン機能等の特徴を持つ。以下、コンパイラの概要と、特徴的言語機能の実現方式について述べ、最後にel(α)のプログラミング環境にも触れる。

2．コンパイラの概要

el(α)コンパイラは、図1に示すように、一般的コンパイラ[2]と同様、構文・意味解析を行うフロント部、副プログラムのインライン展開、及び局所的最適化を行うミドル部、レジスタ割付けを行いながらコードを生成するバック部の3つの部分から構成されている。

フロント部には、構文規則の記述からＬＬ(1)構文解析部生成系により自動生成した構文表を組み込んだ。これを使用して、表駆動方式で構文解析をしながら属性評価をすることによって、意味解析を行う。

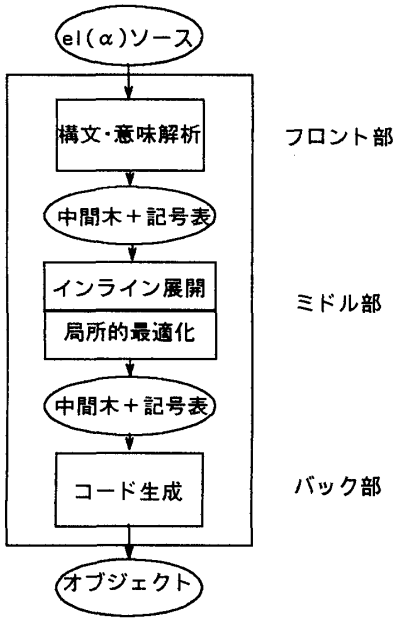

図1 el(α)コンパイラ構成

ミドル部インライン展開処理は、インライン指定のある副プログラムについて呼出し毎に中間形式で展開する。

ミドル部の局所的最適化では、1回目のパスで、最適化の副作用がないように最適化対象となる変数の alias 情報を収集し、ユーザの記述では表れない構造体要素のアドレス計算式を展開する。2回目のパスで基本ブロック単位の共通式削除等の最適化を行う。このパスでは、定数畳み込み、式の構造変換等の式単位の最適化も行う。

バック部では、中間木を再帰下降型で構文解析しながら、基本ブロック単位でレジスタをトレースする方式でレジスタを割付け、オブジェクトコードを生成する。バック部においても冗長な分岐の削除等の最適化を行う。

本コンパイラでは、中間形式として式や文の構造を保持する木構造中間語(以下中間木という)と変数等の記号情報を保持する記号表を使っている。中間木の各ノードは、変数名、配列要素、集合体要素、操作（代入、if、loop,他）等の種別を表現するもので、属性として記号表へのポインタを保持しており、機種に依存しないレベルで定義している。機種に依存する情報は、記号表の限られたフィールド(変数のサイズや割付け場所等)に記録する。ミドル部では、インライン展開も最適化もこの中間形式レベルで処理するので、機種に依存せず、多くの機種にそのまま適用することができる。

このため、本コンパイラは比較的容易に多くの機種に展開可能である。

3． el(α)の機能実現方式

(1)ライブラリの登録・参照

インタフェースユニットは外部から参照可能な宣言を持つコンパイル単位であり、これらを登録したものをライブラリという。外部プログラムからはconnect文で指定することにより参照可能となる。el(α)コンパイラでは、インタフェースユニットを構文･意味解析しながらソースイメージのままライブラリへ登録する。ライブラリを参照しているユニットをコンパイルするときには、ライブラリからconnect文で指定されたインタフェースユニットのソースを入力し、参照ユニットとともに再度、構文･意味解析を行い、整合性のチェックを行う。

Compiler Development of Embeded Computer Language el(α)

Kazuko SAKUMA[†], Tan WATANABE[†], Hideyuki Arai[††], Kohbun UMETANI[†††]

[†]Systems Development Laboratory,Hitachi Ltd.,[††]Odawara Works,Hitachi Ltd.,[†††]HITACH Microcomputer Engineering

(2)副プログラムのインライン展開

el(α)では副プログラムに対してインライン展開が指定可能である。基本的には、フロント部で生成された中間語を展開することで実現する。実パラメタについては、直接、仮パラメタに置換すると再計算の無駄と副作用が起こる可能性がある。そこで、そのまま置換すると実行効率の悪くなる実パラメタにて対してはパラメタ代替用変数を作成することにした。return文については、本体の最後へのgoto文に置換する。図2において、ソースイメージでインライン展開の実現例を示す。

```
var A:record
        P:BOOL;
        Q:array(5) of INT;
      end record;

procedure ONE(X:BOOL; Y:in out INT) inlinable;
do;                                  ｲﾝﾗｲﾝ可能指定
   if X then X:=1 else Y:=0;
end ONE;

procedure TWO;
do;
   ONE(A.P,A.Q(5)) inline;     ―呼出し
end TWO;
```

↓インライン展開

```
procedure TWO;
var $V0:BOOL; $V1:POINT TO INT;      ―テンポラリ
do;                                    変数
   $V0:=A.P;                        ┐前処理
   $V1:=PTR(A.Q(5));                ┘
   if $V0 then $V1^:=1 else $V1^:=0;―展開コード
$LEND : none;                        ―後処理
end TWO;
```

図2 ソースイメージによるインライン展開例

4. 開発結果

コンパイラは、HITAC M シリーズと、機器制御用マシンに対して開発し、機種展開が比較的容易であることを確認した。

実用コンパイラの性能を実測すると、ドライストーンベンチマークでは、最適化なしで 5840 ドライストーン、最適化をを行った場合で 6780 ドライストーンとなり、約16％減の効果があった。

さらに、ドライストーン、バブルソート、その他、制御用プログラム３本について、同じ機能のプログラムをアセンブリ言語で技巧を凝らして書き、オブジェクト実行時間を比較した。これを図３にて示す。最適化なしの場合は、アセンブラ比は、1.1 から 2.2 であり、最適化を行った場合は、1.0 から 1.5 となった。

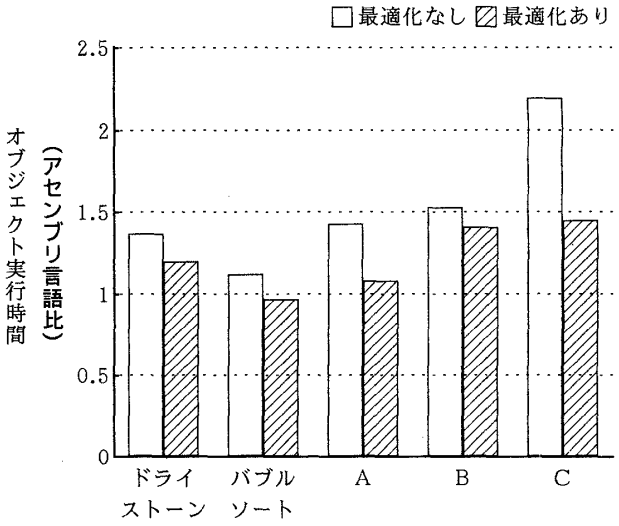

図3 オブジェクト実行時間

5. プログラミング環境

el(α)コンパイラの１つは機器制御用ソフトウエアの開発ツールの一貫として開発した。コンパイラ環境は、図３のように、共同開発において共通テーブルや共通ルーチンの管理を容易にする自動ドキュメント生成ツール、アセンブラプログラムと連結するためのリンケージエディタ、実行時テストを行うためのシミュレータ、等のツールが整備されている。

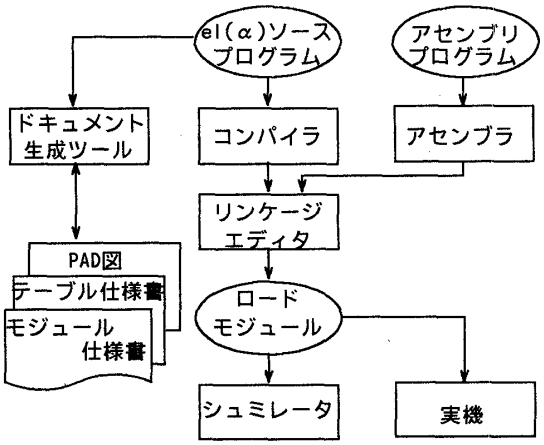

図4 el(α)プログラミング環境

6. おわりに

本コンパイラの開発により、el(α)が他の機種に容易に展開できることを確認した。また、機器制御マシンに適用したコンパイラでは、実用的な性能を得ることができた。

参考文献

[1]渡邊,他:共通的計算機モデルに基づく機器制御用言語el(α)の開発,情報処理学会第40回全国大会予稿集(1990/3)

[2]Aho,A.V., Ullman,J.D.,Compilers Principles, Techniques, and Tools,Addison-Wesley.[1986].