2G－4　コンパイラ内蔵方式による手続きのインライン展開

阿部雄造

富士通株式会社　ソフトウェア事業部

1．はじめに

FORTRAN及びC言語プログラムにおいて，実行性能を向上させる最適化技法として，インライン展開がある．しかし，従来のソースレベルによるインライン展開では種々の制限（文法的な制約，計算式の複雑さ等）があり，充分な性能が得られていなかった．そこで，これを解決する方法として，「コンパイラ内蔵方式によるインライン展開」を考案したので報告する．

2．内蔵方式によるインライン展開の機能

インライン展開とは，手続きの引用（呼び出し）がある場合に，その引用箇所に手続きの本体を直接展開する最適化である．展開することによって，引数や関数値の受渡し及び呼出しと復帰のためのオーバヘッド（レジスタの退避・復元）が削除されるので，実行時間を大幅に短縮することができ，同時に各種最適化も促進される．

FORTRANの副プログラムのインライン展開と最適化が組み合わされたときの波及効果の例を図-1に示す

```
   SUBROUTINE TEST(A,B,C,D,R1,R2)
     REAL A(100),B(100),C(100),D(100)
     REAL NORM
     DO 10 I = 1,100
       A(I) = NORM(C(I),D(I),R1)
       B(I) = NORM(C(I),D(I),R2)
10   CONTINUE
     RETURN
   END
   REAL FUNCTION NORM(X,Y,R)
     NORM = ( X**2 + Y**2 ) / R**2
     RETURN
   END
```

⇩　インライン展開

```
  SUBROUTINE TEST(A,B,C,D,R1,R2)
    REAL A(100),B(100),C(100),D(100)
    REAL NORM
    DO 10 I = 1,100
      A(I) = ( C(I)**2 + D(I)**2 ) / R1**2
      B(I) = ( C(I)**2 + D(I)**2 ) / R2**2
10  CONTINUE
    RETURN
  END
```

⇩　他の最適化

```
  SUBROUTINE TEST(A,B,C,D,R1,R2)
    REAL A(100),B(100),C(100),D(100)
    REAL NORM
    TEMPR1 = R1**2     ·····  不変式の移動
    TEMPR2 = R2**2     ·····  不変式の移動
    DO 10 I = 1,100
      TEMP = C(I)**2 + D(I)**2  ···共通式の除去
      A(I) = TEMP / TEMPR1
      B(I) = TEMP / TEMPR2
10  CONTINUE
    RETURN
  END
```

図-1　インライン展開と最適化（ソースイメージ）

従来のインライン展開は，ソースプログラムでのインライン展開である．この方法では，種々の制限があり，これがインライン展開を阻害する原因となっていた．この制限を緩和する方法として，「内蔵方式によるインライン展開」を考案した．この方式は，副プログラム（FORTRAN）あるいは関数（C 言語）を中間言語レベルでインライン展開する方式である．この方式を採ると，ソースによるインライン展開で制限となっていたものが解決する．それらの例を以下に示す．

①整合配列

二次元以上の整合配列がある副プログラムもインライン展開

Inline Expansion of Procedure by a Method Built in Compiler

Yuzo ABE

FUJITSU,Ltd.

```
PROGRAM MAIN              SUBROUTINE SUB(X,M,N)
 DIMENSION A(2,2)          DIMENSION X(M,N)
 CALL SUB(A,2,2)           DO 10 J=1,N
 PRINT *,A                  DO 10 I=1,M
 END                         X(I,J)=0
                       10  CONTINUE
                           RETURN
                           END
```

②名標

呼び出し元の局所変数名と呼び出し先の外部名の綴りが一致していてもインライン展開

```
PROGRAM MAIN              SUBROUTINE SUB(A)
 CALL SUB(A)               A=FUN(A)
 FUN=A                     RETURN
 PRINT *,FUN               END
 END
```

③引用箇所

引用箇所に関係なくインライン展開

```
PROGRAM MAIN                  FUNCTION FUN()
 IF (..EQ..) A=FUNC( )         .
 DO 10 WHILE(FUN().EQ..)       RETURN
 .                             END
 DO 20 UNTIL(FUN().NE..)
 .
 PRINT *,FUN()
 END
```

④コモンブロック

呼び出し元と呼び出し先のコモンブロックの構成が異なっていてもインライン展開

```
PROGRAM MAIN              SUBROUTINE SUB()
 COMMON /BLK/A,B,C         COMMON /BLK/X(3)
 CALL SUB()                DO 10 I=1,3
 PRINT *,A,B,C              X(I)=REAL(I)
 END                   10  CONTINUE
                           RETURN
                           END
```

ソースで展開するのは，①は，配列の対応をとる時の計算式の複雑さから，却って性能劣化を招き，②～④は，文法的に不可能である．その他，「内蔵方式によるインライン展開」の長所としては，以下のものがある．

①最適化の候補を選出する基となる変数の「出現頻度」が正確に計算される．

呼出し元の変数は，翻訳単位内で何度インライン展開しても同じものを使用するので，「出現頻度」は，手続きの引用回数に比例する．

②組み込まれる副プログラム（関数）で使用される変数名や文番号が有効となる．

呼び出し先の副プログラム（関数）で使用される変数名や文番号（ラベル）は，そのまま呼び出し元に組み込むので，アセンブルリストでも有効となる．従来の方式では，インライン展開によって，生成名や生成文番号に変わるため対応が取り難かった．

③引数と関数の型の組み合わせチェック

引数の組み合わせや，関数の型の組み合わせチェックが行われる．従来の方式では，デバッグ用オプションが指定されたときのみチェックが行われた．

3．性能評価

従来のインライン展開と「内蔵方式によるインライン展開」に於ける実行性能の比較をすると，これまでの実測した範囲内では最大３０％近い（表-1参照）性能向上が見られたものもあり，実運用プログラムの平均を取ってみても３～５％の性能向上が計られている．従って，この新方式は，非常に効果があることが分かる．

表-1　実行時間比較

	従来方式	内蔵方式
ﾌﾟﾛｸﾞﾗﾑ1	1．0	0．82
ﾌﾟﾛｸﾞﾗﾑ2	1．0	0．73
ﾌﾟﾛｸﾞﾗﾑ3	1．0	0．96

注）従来方式を１とした時のCPU 時間比

4．おわりに

実行性能向上を図る上で，「内蔵方式によるインライン展開」と手続きを意識した最適化及びハード性能を最大限に引き出す最適化等を組み合わせて，「内蔵方式によるインライン展開」をより効果的なものとするのが今後の課題である．