219 Compile and Go FORTRANコンパイラの性能評価について

中所武司，渡辺道生，加藤正道，林英治
（日立ソフトウェア工場）

1. 序言

一般のFORTRANコンパイラは，1度コンパイルして得たオブジェクトを何度も実行することを前提としているので，実行の速いオブジェクトの出力が重要である。ところが，プログラミング教育やデバッギングの場合は，出力オブジェクトは1度だけ実行されることが多いので，コンパイルアンドゴー（C&G）のスループットの向上や詳細なエラーメッセージの出力の方が重要である。

このような教育用およびデバッグ用として，われわれが開発し，現在稼動中のCompile and Go FORTRANコンパイラ（FORTCG）について，その処理方式および性能評価について報告する。

2. 言語仕様

JIS 7000を含み，FORTRAN Ⅳにデバッグ文を追加したものであるが，宣言文を実行文の前に置くという制限がある。教育用としてはJIS 7000レベルで十分であるが，一般のプログラム開発のデバッグ用として用いることを考慮して，このような大きな言語仕様にした。

3. 処理方式

本コンパイラの主な基本設計方針は，以下の4項である。

(B1) C&Gスループットの向上 ― (a) コア常駐方式
(B2) 詳細なエラーメッセージの出力 ― (b) 1パスコンパイル方式
(B3) 豊富なデバッグ機能
(B4) 教師のためのオプション機能

(B1)については，(a)および(b)の処理方式によって実現した。すなわち，一般に行なわれているように，各ジョブ毎にコンパイル，エディット，実行を分離して処理することによって生じるOSオーバーヘッドおよびオブジェクトの入出力操作時間を除去するために，本コンパイラは，コアに常駐して，複数のC&Gジョブを連続処理するようにした。また，コンパイルを高速化するために，ソースプログラムを文単位に処理してオブジェクトコードを出力する1パス方式を採った。

コア常駐方式の場合は，ユーザプログラムの実行によってコンパイラが破壊されたり，コントロールがOSに渡ってしまうのを防ぐ処理が必要である。

処理例 (1) 配列要素の引用時には，その範囲のチェックを行う。
(2) 割当て型GOTO文の制御変数の未定義チェックを行う。
(3) エラー割込み処理は，すべて本コンパイラで行う。 etc.

1パス方式の場合は，1パスで完全なオブジェクトを出力できないものは，後処理を必要とするため，これを除去あるいは軽減するような処理が望ましい。

処理例 (1) 宣言文は実行文の前に置く。
(2) 文番号による分岐は，分岐先アドレスエリアを用いて間接分岐する。
(3) 英字名が属性決定前に引用されたとき，チェイン構造で記憶し，属性決定時に充填めをする。 etc.

(B2)については，言語仕様で禁止している事項は，一般にはユーザの責任とされているものもチェックするようにした。

処理例 (1) DOループ内での制御変数，パラメータの変更のチェック。
(2) 手続き引用時の実引数と仮引数の対応の合法性のチェック。
(3) 定数が実引数の時，引用された手続きでの値の変更チェック。 etc.
(B3)，(B4)については，デバッグ文，未定義変数のチェック機能のほか，統計情報収集機能などを備えた。

プログラムの開発にあたっては，1パスコンパイルは，マルチパスに比べ，制御の流れが複雑なので，モジュール化，階層化のほか，Structured Programmingの手法を取り入れて，制御の流れをわかりやすくした。

4. 性能評価

次の3種類のジョブを用いて性能測定を行った。入力カードはディスクにスタックしてジョブを開始し，ディスクに出力した。結果は，弊社現存の一般のFORTRANコンパイラを用いたC&Gジョブとの時間比を右表に示す。

1) 学生ジョブ50個の連続処理（1ジョブ平均ソース29枚）
C&Gスループットは約20倍で，教育用に極めて有効であることを確認した。さらに，このジョブは，コンパイルエラーがあっても実行するモードで処理したので，1度にコンパイルエラーと実行エラーの両方を見つけたものもあり，デバッグ回数を削減している。

2) 科学計算ジョブ（ソース187枚，副プログラム3個）
このような実行時間の長いジョブは，実行速度は0.3倍と遅いが，スループットでは逆に速くなっており，デバッグ用としての有効性を示している。実行時間が遅い理由は，本コンパイラでは，オブジェクトコードの最適化を行っていないこと，配列要素のアドレス計算を実行時に行うことなどによる。

3) STOP・ENDジョブ
このジョブの結果から，OSオーバーヘッド削除の効果が著明である。

以上の1)～3)の結果のように，C&Gスループットの向上を実現した要因は，主に前述した3項であり，それらが全体の削減量の中で占める割合を以下に示す。
(a) コンパイルの高速化（1パスコンパイル方式）------寄与率 20%
(b) 複数仕事続処理によるOSオーバーヘッドの削減------ 〃 40%
(c) エディット操作処理をコラ内で自分で行うこと------- 〃 40%

1) 学生ジョブ50個
- スループット比 21.4倍
- コンパイル速度比 21.8倍

2) 科学計算ジョブ
- スループット比 2.6倍
- コンパイル速度比 8.1倍
- 実行速度比 0.32倍

3) STOP・ENDジョブ
- スループット比 42.2倍
- コンパイル速度比 80.0倍
- 実行速度比 5.3倍

表1. 測定結果

5. 結言

スループットに関しては，満足できる結果が得られた。この種のコンパイラとして，既にWATFOR[1]などがあり，その有効性が確認されている[2]が，本コンパイラは，WATFORと異なる処理方式をとり，その有効性を定量的な評価を行って確めた。今後は，この種のコンパイラのユーザである，学生，教師，プログラム開発者のための付加機能について研究を進めたい。

6. 参考文献

1) D.D. Cowan 他：Design Characteristics of the WATFOR Compiler, SIGPLAN-notice July 1970
2) H.C. Lucus 他：a Method of Software Evaluation : the Case of Programming Language Translator, the Computer Journal vol. 16 no. 3 Aug. 1973