プログラミング-言語・基礎・実践- 11－10
(1993. 3. 11)

超並列計算機用 Fortran コンパイラの設計と試作

大谷 浩司* 小前 晋† 杉森 英夫† 渦原 茂‡ 安村 通晃‡
* 有限会社アックス † 住友金属工業株式会社 ‡ 慶應大学
連絡先:〒 252 藤沢市遠藤 5322 慶應義塾大学環境情報学部安村研究室

キーワード: 超並列, Fortran

1993 年 3 月 11 日

概要

超並列マシンのプログラミングにおいても Fortran 言語に対するユーザの需要は依然高い. しかし, Fortran 言語は非常に古い言語であり, 並列マシン上やベクトルプロセッサ上で動作させる上においても不都合な点も出てきている. 我々は, Fortran90 の仕様を元にした SIMD 型超並列マシン SM-1 上で動作する Fortran である Bee-Fortran を開発している. Bee-Fortran インプリメントに際しては、SAM モデルを考案した。SAM(SIMD Abstract Machine) モデルを採用することにより、Bee-Fortran マシンのアーキテクチャに対し柔軟なものとなっており, 将来のハードウェアの変更や他のマシンへの対応が容易になっている. 本稿では Fortran を超並列マシンで動作させる場合の問題点と共に SAM モデルおよび Bee-Fortran の仕様やインプリメントについてのべる.

Design and Implementation of a Fortran Compiler for Massively Parallel Machines

Koji Ohtani* Susumu Komae† Hideo Sugimori†
Shigeru Uzuhara‡ Michiaki Yasumura‡
* Axe, Inc. † Sumitomo Metal Industries, Ltd. ‡ Keio University.
Address:Keio University Yasumura Lab. 5322 Endo Fujisawa shi.

Keyword:Massively parallel, Fortran

March 11, 1993

Abstract

Fotran has been a popular programing language. Many users want to use Fortran on programing on massively parallel machines. However it has some shortcommings on programming for vector or massively parallel machines. We have developed Bee-Fortran , a Fortran compiler for SIMD type massively parallel machine, SM-1. The specification of Bee-Fortran is based on Fortran90. We think out SAM(SIMD Abstract Machine) model in the implementation of the Bee-Fortran. SAM model makes Bee-Fortran flexbile to implement on the improved SM-1 in future or other machines. In this paper we describe issues when implementing Fortran to massively parallel machines. We also describe the specification of Bee-Fortran, SAM model and the implementation of Bee-Fortran.

1 はじめに

超並列マシンのプログラミングにおいてもFortran言語に対するユーザの需要は依然高い。しかし，Fortran言語は非常に古い言語であり，並列マシン上やベクトルプロセッサ上で動作させる上においても不都合な点も出てきている．我々は，Fortran90の仕様を元にしたSIMD型超並列マシンSM-1[1]上で動作するFortranであるBee-Fortranを開発している．

Bee-Fortranは次のような特徴をもつ．

1. Fortran-90[3]などと互換性がある．

   Fortranでは，特に互換性が重要となるが，Bee-Fortranで拡張された部分は次世代のFortranであるFortran90の配列操作の仕様と互換性がある．

2. マルチターゲット

   一度作成したアプリケーションは多くの場所で使用できるのが望ましい．しかし，標準となるマシンは存在しないし，ハードウエアは改良され続ける．そのため，Bee-FortranのターゲットはSM-1であるが様々なマシン上に移植するのが容易な構造にして，いろいろなマシン用のオブジェクトを出せるようにした．

3. SAM(SIMD Abstract Machine)

   前項の目的のために，SIMDのマシンの一般的な特徴を抽象化したマシンSAMを考え，SAMをターゲットとする層を設けることにより効率を低下させずに移植性を高めた．

4. C言語をターゲットとする．

   C言語の最適化は近年著しく進歩している．そのため，ターゲット言語をC言語としても，アセンブラをターゲットにした場合と比較しても効率の低下はあまりない．そのため，アセンブラではなくC言語をターゲットにすることによって移植性を高めると同時に最も低レベルの最適化をC言語に任せることによりシステムの肥大化を避けた．

   SM-1上では，並C[2]と呼ばれるC言語を拡張したものが開発されており，Bee-Fortranのターゲット言語としても並Cを採用した．

以下では，まず，ターゲットマシンであるSM-1のアーキテクチャと並Cについて簡単に述べる．次に，Bee-Fortranの言語仕様について説明した後Fortran言語をSIMD型超並列マシンに実現する際の言語仕様上の問題点を述べる．その後，SAMについて説明をし，Bee-Fortranのインプリメントについて述べる．

2 SM-1

SM-1[1]はSIMD型の超並列マシンであり図1に示すシステム構成をしている．マシンはフロント・エンド・プロセッサ(Front End Processor. 以下ではFEと略する.)部とプロセッサ・エレメント(Processor Element. 以下ではPEと略する.)アレイ部にわかれる．PEは1024個備え，32×32のメッシュ状に配置されて隣あったPEは通信路によって接続されている．(NEWS型通信路) メッシュの端の部分はトーラス状あるいは渦巻状に接続することができ，1024×1の直線状に配置されたPEとしても使用できる．また，シャッフル・エクスチェンジ(shuffle-exchange)型の通信路も備え，PE上のデータのリダクション演算などに効果を発揮する．FEから全PEへデータをブロードキャストすることができ，逆にPE上のデータをORをとってFEに戻すことができる．FEは，CPUにSPARCを使用したワークステーションであり，PEアレイ部はSPARCのコプロセッサとして動作する．PEの命令体系は32ビットの3オペランド型となっており，整数の加減乗除およびIEEEの浮動小数点数の加減乗除が可能である．ただし，浮動小数点の演算用のハードウエアは備えておらず，演算はマイクロプログラムによって実行される．各PEは，アクティビティ状態をもち，アクティブでないPEは演算に参加しない．

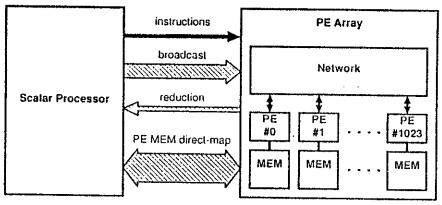

図 1: SM-1 のシステム構成

3 並 C

並 C[2] は，SM-1 用の拡張された C 言語であり，SM-1 上のプログラミングの基本となる．並 C は，他の言語のターゲット言語となることを意識しており，プログラマは PE を直接に認識できるようになっている．PE 上のデータは，パラレル・クラスのデータとして用意され，既存のデータすなわち FE 上のデータはスカラー・クラスとなる．パラレル・クラス変数間の演算，スカラー・クラスとパラレル・クラスのデータ間の演算を行なうことができる．また，制御文として，PE のアクティビティを制御する文，pif, pwhile, pfor, pdo-while をもつ．PE 間の通信は，後置演算子として用意されている．FE から PE へのブロードキャストは，スカラー・クラスからパラレル・クラスへの型変換であり暗黙的に行なわれる．

4 言語仕様

並列マシンやベクターマシンで Fortran を動作させる場合には，いくつか方法がある．

ひとつは，ライブラリによるものである．これは，主に並列マシンで行なわれる方法であり，通信や同期などを行なうライブラリを用意してプログラマが陽に指定するようにしたものである．コンパイラに加える変更も少なくて済むうえ，プログラマが制御を細かく指定できるので効率の良いプログラムを書くことができる．しかし反面，プログラマに対する負担が大きくプログラマの生産性を低下させ，効率のよいプログラムを書くことは非常に困難である．また，既存の Fortran のプログラムを動作させるには大きな変更を加えなければならず，またアルゴリズムやプログラムの方法も大きく変わることもあるためにユーザには受け入れられにくい．

もうひとつは，既存の言語仕様のままコンパイラによって自動的にベクター化や並列化を行なう方法である．この方法は，既存のプログラムを変更せずに動作させることができ，プログラマも新たな仕様を学習しなくても良いため，ベクターマシンにおいて大きな成功を収めた．しかしながら既存の言語仕様のみではプログラマの意図をコンパイラが知るのは非常に困難な場合があるために，並列化やベクター化の結果には不満が残るものであった．そのため，コンパイラ・ディレクティブを使用することによってプログラマがコンパイラにその意図を知らせて並列化，ベクター化を助けるように拡張されていた．このようなディレクティブの導入は，プログラマが学習しなくても良いという利点を減殺している．

最後は，言語仕様自身を拡張するものである．ディレクティブという新たなものを導入するのならば，Fortran の本来の部分とディレクティブによる指定という二重の指定という無駄をなるべく省くように言語仕様を拡張してもプログラマに対する負担はあまり増えない．むしろ，ディレクティブで指定するよりも言語自体で表現した方が自然でありプログラマの負担は軽い．ディレクティブではマシンに依存することを指定する．言語仕様の拡張は，自動的な並列化，ベクター化を妨げるものではなく，同時に行なって良い．そのため我々もこの方法を採用することにした．

言語仕様の拡張はいくつか行なわれてきている．

CM-Fortran[4] は，超並列マシン CM-2[5] のために作成された Fortran であり，配列を並列に処理するために配列に関する操作の拡張，制御構造の追加，内部関数の追加，新たなディレクティブの定義を行なっている．

Fortran90 は Fortran77 に次ぐ次世代の Fortran として定義されているものである．ポインタや構造体，動的な記憶領域確保，モジュール

などの拡張に加えてCM-Fortranと同様な配列に関する操作の拡張，制御構造の追加，内部関数の追加が行なわれている．

HPF(High Performance Fortran)[6]は並列マシンのためにFortran90をもとにディレクティブ，制御構造を追加している．

Fortranは歴史の長い言語であり，長期間多くの人々によって使用されてきた．従って，言語仕様の決定に関しては互換性の問題が重要となる．そのため，我々は次世代のFortranの仕様として決定されつつあるFortran90の配列に関する拡張を元にすることにした．それに，CM-Fortranを参考にしてディレクティブを加えた．

4.1 Bee-Fotranの言語仕様の概略

4.1.1 全称配列と部分配列

Bee-Fortranでは式中で配列のすべての要素や要素の部分集合を参照することができる．これらはFortran90の仕様に準拠している．

1. 全称配列

   添字を伴わない配列はすべての要素の集合を表わす．これを全称配列と呼ぶ．例えば，配列A(10)が存在している時，全称配列はAで表現される．

2. トリプレット添字

   次のように':'で区切って三つの整数式を書くことにより添字の集合を表わすことができる．

   top:bottom:stride

   これは，topからbottomまでstride毎の添字の集合をあらわす.top, bottom, strideは省略することができ，省略時の値は各々添字の最小値，添字の最大値，1となる.strideを省略するばあいは，二つめの':'も省略して良い．

   例)
   2:10:2は，添字の集合{2, 4, 6, 8, 10}を表わす．':'のみは，添字の全集合を表わす．

3. ベクター添字

   添字に整数のベクター(一次元の配列)を指定しても良い．ベクターを指定した場合は，要素の値の集合が添字の集合になる．

4. 部分配列

   添字にトリプレット添字またはベクター添字を指定した場合は，配列の要素の部分集合をあらわし部分配列と呼ぶ．

4.1.2 配列代入文

配列を四則演算や代入文の左辺に指定することができ，その場合は対応する要素毎に演算が行なわれる．また，例えばスカラーと配列の演算も指定できその場合は，配列の全要素とスカラー値の演算を行なわれる．たとえば，

```
      DIM A(10), B(10), C(10)

      A = B + C + 2
```

は，

```
      DIM A(10), B(10), C(10)

      DO 10 I = 1,  10
          A(I) = B(I) + C(I) + 2
10    CONTINUE
```

と同じ結果が得られるものとする．しかし，各要素の演算の実行の順序は規定しない．

4.1.3 WHERE構文

配列代入文においてマスクを指定して演算の行なわれる要素を制限することができる．この文は次の形をしている．

```
WHERE (マスク式)
    配列代入文1
ELSE WHERE
    配列代入文2
END WHERE
```

マスク式の結果は論理型の配列であり，マスク式の結果がTRUEである要素と対応する要素についてのみ配列代入文1が実行される．マスク式の結果がFALSEである要素と対応する要素のみ配列代入文2が実行される.ELSE WHERE以降はなくても良い．

4.1.4 その他

その他に，INTERFACE文，配列構築子や内部関数を拡張している．

5 言語仕様上の問題点

Fortranの言語仕様には，超並列マシン上では実現する際に問題となる点がふたつある．一つはシーケンス・アソシエーション (Sequence association) の問題であり，もう一つは，ストレージ・アソシエーション (Storage association) の問題である．

次のような副プログラムがあったとしよう．

```
SUBROUTINE FOO(A, N, M)
REAL A(N, M)
INTEGER N, M
...
END
```

Fortranでは，この副プログラムを次のように呼ぶことができる．

```
REAL B(10, 20, 30)

CALL FOO(B(5, 1, 1), 15, 40)
END
```

つまり，ある配列を次元数や各次元の大きさの異なる配列引数として指定できるのである．しかも，その先頭も変更できるし，次元数，各次元の大きさも実行時に決定できる．このことは，Fortran90では次のように定義されている．

配列の全要素は配列要素順 (array element order) によって順序づけられたシーケンスを構成する．副プログラムの配列の実引数と仮引数の対応は，このシーケンス同士の対応となる．すなわち，仮引数のある要素には，その要素のシーケンス中の順番と同一の順番を実引数のシーケンス中で付けられている実引数の要素と対応づけるのである．これをシーケンス・アソシエーションと呼ぶ．配列の要素を実引数で示した場合は，配列全体のシーケンスのうちその要素以降のシーケンスを示す．実引数のシーケンスが

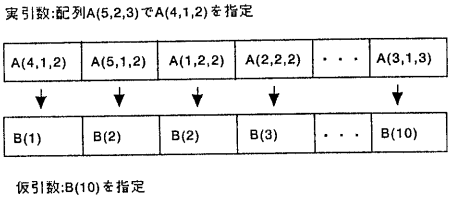

図 2: シーケンス・アソシエーション

仮引数のシーケンスよりも長い場合は余分は捨てられる．(図2参照)

配列要素順は，スカラープロセッサにおいてメモリに割り付けられる順と一致するために，シーケンス・アソシエーションはスカラープロセッサでは，配列の先頭または，指定の要素のアドレスを副プログラムに渡すことによって簡単に実現できる．しかし，SM-1のようにメモリが分散しており，配列の要素を分散したメモリに配置する場合には大きな問題となる．

シーケンス・アソシエーションを分散メモリで実現する方法には，次のふたつが考えられる．

1. 副プログラムの先頭では，実引数のアドレスや配置の情報だけを受け取り，実際に必要な場所でこれらの情報を用いて要素にアクセスする．

2. 配置が異なる場合は副プログラムでの配置に要素を複写する．Fortranでは，引数は参照による呼びだし (call by reference) なので，副プログラム中での内容の更新を反映しなければならず，副プログラム実行後にもとの配列に複写して戻す必要がある．

最初の方法には，必要でない要素の複写が生じない利点があるが，要素をアクセスする複雑なコードが頻繁にコード中に出現し，実行時の効率はあまり良くないだろう．しかも，複写が必要でない場合でも，こういったコードが必要になる．ふたつめの方法は，複写をいつ行なうかでさらにふたつの方法にわかれる．すなわち，複写を副プログラム側で行なう方法と呼び出し側で行なう方法である．複写を副プログラムで行なう場合は，呼び出し側では何らの手続きの

必要はなく実引数の情報のみを副プログラムに渡せば良い．これは，最も実現が容易な方法でもあるが，副プログラムはどのようにして呼び出されるかを知ることはできないために複写のためのコードは常に必要となる．呼び出し側で複写する場合は，副プログラム側には余分なコードは必要がなく，複写が不必要な場合はそのための無駄は一切ない．また，呼び出し側の文脈による複写の最適化を行なえることもある．しかし，Fortran77 の仕様では呼び出し側では，副プログラムの引数の形を知ることができないため，複写のコードを出すことができない．

Bee-Fortran では，最後の方法を採用した．副プログラムの引数の形を呼び出し側に知らせるため Fortran90 の仕様にある INTERFACE 文を導入し，実引数と形が異なる場合は，INTERFACE 文が必要であることとした．実引数と形が異なることはそれほど多くないため受け入れられ得ると考えられる．

ストレージ・アソシエーションの問題は，COMMON 文や EQUIVALENCE 文で生じる．次のような文があったとする．

```
COMMON /FOO/ A(10), B, C
REAL A(10), B
DOUBLE PRECISION C
```

また，異なるモジュールに次の文があったとする．

```
COMMON /FOO/ V, W(5), X(5), Y, Z
REAL V, W(5), X(5), Y, Z
```

REAL 型は 32bit，DOUBLE PRECISION 型は 64bit で表わされているものとする．このばあい，V は A(1) と同じ記憶領域を占め，Y は C の内部表現の前半 32bit と Z は後半 32bit と同じ領域を占める．このように Fortran では，同じ名前の COMMON 文は変数や配列が異なっても同じ記憶領域を占める．これをストレージ・アソシエーションと呼ぶ．

スカラープロセッサでは，これはコンパイル時のアドレスの割りつけだけの問題であり，実現も容易である．しかし，配列を分散したメモリに割り当てようとする場合は大きな問題となる．なぜなら，アドレスだけの問題ではなく，プロセッサの割り当ての問題も生じてくるからである．これは，他のモジュールでの COMMON 文を知ることなしには決定できない．

このような COMMON 文のうち，一つの変数や配列の記憶領域を複数の変数や配列と共有するものは，あまり多くない．そのため，この機能の実現に資源を費やしても利点は少ないと考え，Bee-Fortran では PE の記憶領域にこのような COMMON 文の領域を割り当てることは言語仕様上禁止することとした．

6 SAM(SIMD Abstract Machine) モデル

マルチターゲットを実現し，ハードウェアの変更に対し柔軟にするために抽象的な SIMD マシンのモデルを導入した．Bee-Fortran 処理系では，まず，この抽象マシンをターゲットとした中間表現にプログラムを変換する．次に最適化の後，この中間表現から実際の各マシン用にオブジェクトを生成する．この抽象的なマシンを SAM(SIMD Abstract Machine) と呼び，SAM をターゲットとする中間表現を SAM 中間表現と呼ぶ．また SAM をターゲットする変換を SAM 化と呼ぶことにする．

SAM は，任意個の数のプロセッサ・エレメントとフロントエンド・プロセッサから構成される．このプロセッサ・エレメントを APE(Abstract Processor Element) と呼ぶ．APE は 2 次元格子状または 1 次元の列状に配置される．隣どうしに配置された APE 同士の通信は離れた APE よりも高速である．各 APE はアクティビティを持ち，アクティブなプロセッサのみが動作する．

SAM では APE の数が任意であるため，配列のデータを割り当てる際に折り返しを考える必要がない．従って 1 次元または 2 次元の配列の場合は，1 要素を一つの APE に割り当てる事ができる．1 次元の場合は直線上の APE に，2 次元の場合は格子上の APE にそのまま割り当てる．例えば A(100, 100) は 100 × 100 の格子上の APE に割り当てる．しかし，一般には一つの APE には配列の複数の要素が割り当てる．例えば三次元配列の三番目の次元は同一

のAPEに割り当てる．配列A(2, 2, 2)の要素A(2, 1, 1)とA(2, 1, 2)は同一のAPEに割り当てる．以上は，現在採用している割当の場合であるが，その他の割当の場合も，APEこのモデルを使用する事ができる．例えば，隣接したいくつかの要素を同一のプロセッサに割り当てるブロック割当で，A(40, 40)を割り当てる場合，ブロックサイズを10とすると，4×4のAPEに割り当て，各APEには100個の要素を割り当てる．フロント・エンド上に割り当てる配列の場合も，これの特殊な場合と考えられる．すなわち，全ての要素を唯一のプロセッサに割り当てるだけである．

このモデルの利点は以下の点である．

- 実際のアーキテクチャに近く通信のオーバヘッド等を考えやすい．
- 実際のプロセッサの数に依存しない．
- インプリメントが容易である．
- フロントエンド・プロセッサ上の配列も効率良く処理でき，スカラープロセッサにも対応できる．

7 インプリメント

7.1 処理系の構成

Bee-Fortran処理系は，Fortran77をC言語に変換する処理系f2c[7]を元に作成した．図3にBee-Fortran処理系の構成を示す．枠に囲まれているのは，処理系中のコンポーネントを表す．まず，処理系に入力されたBee-Fortranソースコードは，パーサーが解析し中間表現に変換する．次に中間表現を解析して配列の割り当てを決定する．中間表現は，前処理の後，SAM化を行ないSAM中間表現となる．その後，必要に応じて最適化を行う．最後にSAM中間表現からターゲット言語を出力する．

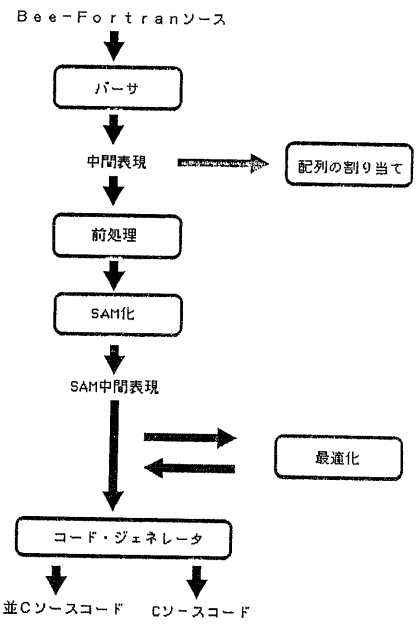

図 3: 処理系の構成

7.2 SAM化

SAM化をどのように行なうかを説明する．SAM上で配列の演算を行なうためには次のことを行なう必要がある．

1. アクティビティの制御

   不必要な演算が実行されないように，配列の形とマスクによってAPEのアクティビティを制御しなければいけない．

2. 演算を行なうAPEへのデータの移動

   演算を行なうには，演算の対象となるデータを同一のAPEに置く必要がある．

3. APE内のデータの選択

   APE内には複数のデータが置かれているため配列の演算では同一の演算をこの複数のデータに施す必要がある．

以上を実現するのにデータの移動をいつ行なうかによって二つの方法が考えられる．ひとつは，これらを各データの演算の直前で個々のデータ毎に行なう方法である．この方法によると不必要なデータの移動を行なわないようにするのは非常に容易であり，インプリメントも簡単である．しかし，演算に比べて通信に時間を要するプロセッサが一般的であるため演算の間に通信

を入れることは演算のパイプラインを乱し効率が良くない．もうひとつは，配列の最初の演算に入る前にすべてのデータの移動を行なう方法である．この方法では，多くのデータの移動を行なうため全体としてより良い移動の戦略を採用することができる．また，移動のためのオーバヘッドは全体として個々に行なうよりも小さくできる．

以上のことからBee-Fortranでは，データの移動は演算に入る前に移動を行なう方法を採り，配列代入文などは以下のように実行する．

1. 一時変数の確保
2. データの移動
3. マスクや配列の形によるアクティビティの決定
4. 決定したアクティビティによるAPEの選択

   アクティブにすると決定したすべてのAPEを実際にアクティブにして以下を行なう．
5. APE内のデータの選択

   演算に関係のあるデータすべてについて次を行なう．
6. 選択したデータによる演算

7.3 コード・ジェネレーション

コード・ジェネレーションではSAM化された中間表現を参照してターゲット言語を生成する．ターゲットのマシンでは実際のPE上に複数のAPEを配置することになる．SAMモデルの目的からAPEの配置は実際にも近い形でPEに配置しなければならない．SM-1では格子の縦にならんでいるAPEはPE上でも縦の方向に並べる．そして，縦のAPEの数がPEの数を越えた場合は，同じ縦列の最初のPEに戻って配置する．横も同様である．

PE内にAPEが複数存在することからアクティビティによるAPEの選択は次のようになる．

> PE内のすべてのAPEについて調べてアクティブなAPEの時だけ実際にPEをアクティブにして実行する．

8 おわりに

我々が開発中のBee-Fortranについてその仕様，インプリメンテーションについて述べてきた．現在，処理系はSM-1上の並C言語およびスカラー・マシン上のK&R C言語をターゲットとしてほぼ動作中である．最適化および他の超並列マシンをターゲットとしたインプリメントが今後の課題である．

最後に，並CおよびSM-1の開発を推進された湯淺太一助教授をはじめとする豊橋技術科学大学湯淺研究室の皆さん，住友金属工業のSM-1開発に携わった皆さんに深謝します．

参考文献

[1] 松田元彦, 湯淺太一, SIMD型超並列計算機SM-1(仮称)の概要とその性能, 情報処理学会研究報告, 92-ARC-87 August, 1992.

[2] 貴島寿郎, 湯淺太一, SIMD型超並列プログラミング言語「並C」とそのコンパイラ, 情報処理学会研究報告, 92-PRG-9, October, 1992.

[3] ISO, Fortran 90, May 1991. ISO/IEC 1529:1991 (E).

[4] Thinking Machines Corporation, CM Fortran Reference Manual, September 1989, Version 5.2-0.6.

[5] ダニエル・ヒリス著, 喜連川 優訳, コネクション・マシン, September 1990, パーソナル・メディア

[6] Digital Equipment Corporation, High Performance Fortran Proposal, January 27, 1992.

[7] S.I.Feldman et al., A Fortran-to-C Converter, AT&T Bell Laboratories Computing Science Technical Report NO.149, August 2, 1990.

[8] JIS Programming Language FORTRAN, JIS C 6201-1982.