software AG

Natural for Mainframes

Natural Optimizer Compiler

バージョン 4.2.5

October 2009

Natural

This document applies to Natural バージョン 4.2.5 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

目次

1 Natural Optimizer Compiler

Natural Optimizer Compiler のこのドキュメントでは、Natural Optimizer Compiler がサイトにインストールされている場合に考慮する必要があるさまざまな側面について説明します。

Natural Optimizer Compiler ドキュメントでは、Natural Optimizer Compiler は、製品コードである NOC とも呼ばれます。

このマニュアルで使用されるフォーマットの省略形については、Naturalの『ステートメント』ドキュメントの「フォーマット」セクションを参照してください。

全般的な情報	Natural Optimizer Compiler のさまざまな側面および Natural Optimizer Compiler の活用方法。
Optimizer Compiler のインストール	Natural Optimizer Compiler のインストール。
Optimizer Compiler の使用	コンパイルに使用されるステートメントおよびプログラム。 Natural Optimizer Compiler での処理に適しているプログラムの統計データ：`NOCSTAT` コマンド。 Optimizer Compiler を使用する場合の例。
Optimizer Compiler のアクティブ化	Natural Optimizer Compiler をオンに切り替える方法。
Optimizer オプション	Natural Optimizer Compiler のさまざまなオプション。 `PGEN` を適用して、生成されたコードおよび内部Natural構造を調査のために出力する方法。 他の Natural パラメータの影響。
パフォーマンスの考慮事項	データフォーマット、配列、英数字フィールド、`DECIDE ON`、および数値を考慮して、最高のパフォーマンスを得る方法。
Zap のリスト	Natural Optimizer Compiler に適用されている Zap の概要を取得する方法。

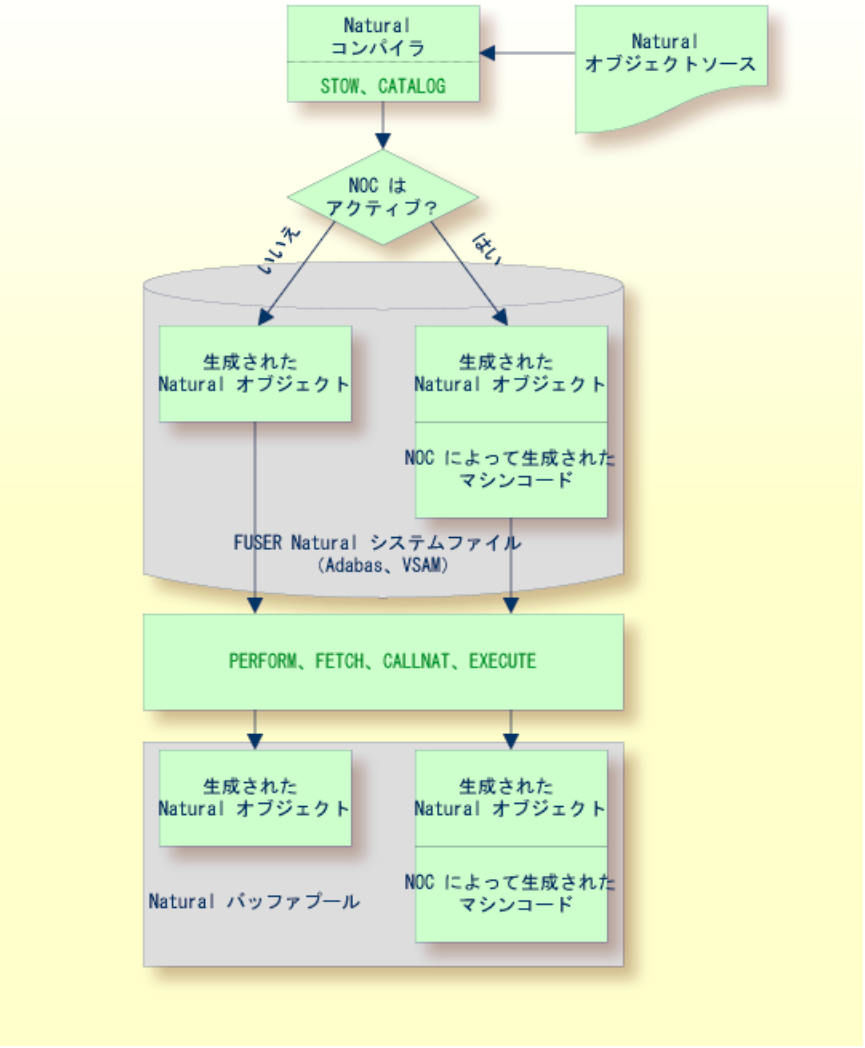
2 NOC - 全般的な情報

このセクションでは、Natural Optimizer Compilerがサイトにインストールされている場合に考慮する必要があるさまざまな側面について説明します。 このドキュメントに記載されている情報は、Natural Optimizer Compiler によって提供される利点を十分に使用する場合に役立ちます。

Natural ニュークリアスの最適化

Naturalニュークリアスによって、単純な算術演算、割り当て、および比較ステートメントが最適化されます。この最適化は、これらの一部をマシンコードに変換することによって行われます。 すべてのプログラムがこの方法で自動的に最適化されます。

次の図は、Natural プログラミングオブジェクトがコンパイルまたは実行されるときの Natural Optimizer Compiler によるマシンコードの生成方法を示しています。



Natural Optimizer Compiler

Natural Optimizer Compiler は、標準的な最適化よりも一歩進んでいます。 単純なステートメントだけでなく、複雑なステートメントやステートメントシーケンスもマシンコードにコンパイルされます。

コンパイルされたコードは、配列範囲処理、フィールド連結、および最適なベースレジスタ割り当てに関して、さらに最適化されます。

NOC で最適化されたすべてのステートメント（算術演算を含む）は、標準 Natural で生成された同じステートメントと同じ結果を生成します。

Natural Optimizer Compiler をアクティブ化するには（関連セクションを参照）、Natural パラメータモジュールのマクロ NTOPT、ダイナミックプロファイルパラメータ OPT、システムコマンド NOCOPT、または OPTIONS ステートメントを使用します。

アクティブ化された Natural Optimizer Compiler でカタログされる（STOW または CATALOG システムコマンド）すべてのプログラムは、マシンコードにコンパイルされます。 そのため、プログラムをどの程度最適化できるかに応じて、プログラムのオブジェクトコードサイズが通常よりも大きくなるという結果も生じます。

Natural Optimizer Compiler がプログラムのすべてまたは一部に対してシステムコマンド NOCOPT、マクロ NTOPT、または OPTIONS ステートメントでアクティブ化された場合、RUN システムコマンドで実行されたプログラムは、マシンコードにコンパイルされます。

プログラムが Natural Optimizer Compiler によるコンパイルに適しているかどうかを確認するには、関連セクションで説明されている NOCSTAT コマンドを使用します。

注意: マシンコードにコンパイルされたプログラムには、ダイナミックな再カタログ機能（プロファイルパラメータ RECAT を ON に設定）は使用できません。

Natural Optimizer Compiler でコンパイルされたプログラムを実行する場合、Natural Optimizer Compiler がインストールされている必要はありません。

3 Natural Optimizer Compiler のインストール

この章では、サポートされているさまざまな環境での Natural Optimizer Compiler（NOC とも呼ばれる）のインストール方法について説明します。

全般的な情報

インストールジョブ

Software AG 製品のインストールは、インストールジョブによって実行されます。 ジョブは、手動で作成するか、Software AG の System Maintenance Aid（SMA）によって生成されます。

以下で説明する各インストール手順では、対応するタスクを実行するジョブのジョブ番号が示されています。 ジョブ番号は、SMA によって生成されたインストールジョブに対応しています。

System Maintenance Aid の使用

インストール処理での SMA の使用については、『System Maintenance Aid』ドキュメントを参照してください。

必要条件

製品およびバージョンは、Natural の最新のリリースノートの「*Natural および他の Software AG 製品*」と「*必要なオペレーティング／TP システム*」に指定されています。

インストールテープ - z/OS

インストールテープには、以下の表に挙げられているデータセットが収録されています。

データセット名	内容
NOC*vrs*.LOAD	このデータセットには、Natural Optimizer Compiler ロードモジュールが含まれています。

データセット名の *vrs* は、製品のバージョン、リリース、およびシステムメンテナンスレベル番号を表します。

インストールテープの詳細については、テープに付属の「*テープ作成レポート*」を参照してください。

スペース要件

データセットがディスク上で必要とするスペースについては、「*テープ作成レポート*」を参照してください。

インストールテープ - z/VSE

インストールテープには、以下のデータセットが収録されています。

データセット名	内容
NOC*vrs*.LIBR	LIBR バックアップファイル。

データセット名の *vrs* は、製品のバージョン、リリース、およびシステムメンテナンスレベル番号を表します。

インストールテープ - BS2000/OSD

インストールテープには、以下のデータセットが収録されています。

データセット名	内容
NOC*vrs*.MOD	Optimizer Compiler モジュールライブラリ。

データセット名の *vrs* は、製品のバージョン、リリース、およびシステムメンテナンスレベル番号を表します。

インストールテープの詳細については、テープに付属の「*テープ作成レポート*」を参照してください。

スペース要件

データセットがディスク上で必要とするスペースについては、「*テープ作成レポート*」を参照してください。

インストールテープ - VM/CMS

インストールテープには、以下の表に挙げられているデータセットが収録されています。

データセット名	内容
NOC*vrs*.TAPE	このデータセットには、Natural Optimizer Compiler ロードモジュールが含まれています。

データセット名の *vrs* は、製品のバージョン、リリース、およびシステムメンテナンスレベル番号を表します。

インストールテープの詳細については、テープに付属の「*テープ作成レポート*」を参照してください。

スペース要件

データセットがディスク上で必要とするスペースについては、「*テープ作成レポート*」を参照してください。

ディスクへのテープ内容のコピー

手順 3.1. テープ内容をディスクにコピーするには

1 以下の式に従ってテープマーク数を計算して、TAPE LOAD コマンド用の位置までテープを巻きます。

「*テープ作成レポート*」に示されているように、NOC*nnn*.TAPE のシーケンス番号が *n* であるとすると、3*n*-2 テープマーク（つまり、最初のデータセットは FSF 1、2 番目は FSF 4、以下同様）の位置に移動する必要があります。

2 Natural インストールファイルを保存するディスクにディスク A としてアクセスします。

3 システムオペレータに依頼して、仮想マシンのアドレス X'181' にテープドライブを接続し、Natural Optimizer Compiler インストールテープをマウントします。

4 テープが接続されたら、以下の CMS コマンドを入力します。

```
TAPE REW
```

以下の CMS コマンドを入力して、テープを目的の位置まで巻きます。

```
TAPE FSF n
```

n は、上記で計算（3*n*- 2）したテープマーク数です。

5 以下の CMS コマンドを入力して、Natural Optimizer Compiler/CMS インストールマテリアルをロードします。

```
TAPE LOAD * * A
```

テープは後のインストール手順で必要であるため、テープドライブは仮想マシンに接続したままにします。

インストール手順

手順 **1 - Natural** パラメータモジュールの変更 **-** ジョブ **I060**、**I080**

Natural パラメータモジュール（NATPARM）に以下のマクロを追加して、Natural Optimizer Compiler をアクティブ化します。

```
NTOPT ON
```

パラメータモジュールをアセンブルおよびリンクします。

手順 **2 -** すべての **Natural** ニュークリアスの再リンク **-** ジョブ **I060**、**I080**

ニュークリアスのリンク手順を変更します。

■ **z/OS**
以下の INCLUDE 命令を Natural ニュークリアスのすべてのリンクに追加します。共有ニュークリアスを使用している場合は、このステートメントを共有部分のリンクに含めます。

```
INCLUDE NOCLIB(NOCNUC)
```

対応する DD ステートメントを追加します。

```
//NOCLIB DD DSN=NOCvrs.LOAD,DISP=SHR
```

- **z/VSE**
  以下の INCLUDE 命令および Natural Optimizer Compiler の対応するサブライブラリを、リンケージエディタの検索チェーンに追加します。

```
INCLUDE NOCNUC
```

- **BS2000/OSD**
  以下の INCLUDE 命令を、NATvrs.JOBS の LNATSHAR 要素に追加します。

```
INCLUDE NOCNUC,NOCvrs.MOD
```

  Natural ニュークリアスを再リンクします。Natural の『*インストール*』ドキュメントの「BS2000/OSD 環境での Natural のインストール」の「*手順9：Natural ニュークリアスのリンク*」に説明があります。

- **VM/CMS**
  Natural モジュールまたは DCSS にインクルードされるテキストファイルのリストは、REXX プログラム NAT$LOAD EXEC（変数 LOADLIST）に含まれています。 Natural システムをカスタマイズするには、XEDIT を使用してこの EXEC の LOADLIST を必要に応じて変更します。

  以下の INCLUDE 命令を、プログラム NAT$LOAD EXEC に追加します。

```
LOADLIST = LOADLIST 'NOCNUC'
```

  Natural ニュークリアスを、プロシージャ NATBLDM に再リンクします。

インストール確認

1. 既存のプログラムを再カタログするか、新しいプログラムを作成してカタログします。
2. LIST システムコマンドを使用して、カタログしたプログラムのディレクトリ情報をチェックします。

```
LIST DIR object-name
```

指定したオブジェクトのディレクトリ情報が表示され、マシンコードのサイズが画面の一番下に表示されます。

4 Optimizer Compiler の使用 - 概要

- コンパイルの対象および対象外
- **NOCSTAT** コマンド
- マシンコードのサイズの表示
- **Optimizer** の使用例

5 コンパイルの対象および対象外

Natural Optimizer Compiler は、計算、転送、論理条件処理など、大量のデータ操作が含まれるプログラムに特に効果的です。

Natural Optimizer Compiler では、以下のステートメントがマシンコードにコンパイルされます。

- 割り当てステートメント（ASSIGN および MOVE）
- RESET
- 算術ステートメント（COMPUTE、ADD、SUBTRACT、MULTIPLY、DIVIDE）
- 条件ステートメント（IF、DECIDE）
- 制御ステートメント（FOR、REPEAT）
- ESCAPE
- COMPRESS
- EXAMINE

  以下の節のみを使用できます。

  GIVING NUMBER、GIVING POSITION、または GIVING LENGTH（Natural の『ステートメント』ドキュメントも参照）。

GIVING INDEX は最適化されません。 例：

```
EXAMINE #TEXT FOR #A GIVING NUMBER #NMB1
EXAMINE #TEXT FOR #A GIVING POSITION #POSEX5
EXAMINE #TEXT FOR #A GIVING LENGTH #LGHEX6
```

Natural Optimizer Compiler では、以下のステートメントはコンパイルされません。

- I/O ステートメント（DISPLAY、WRITE、READ/WRITE WORK FILE）。
- 複雑な特殊ステートメント。SEPARATE など。
- 別のプログラミングオブジェクトに制御を渡すステートメント。FETCH、PERFORM、CALLNAT、CALL など。
- データベースアクセスを実行するステートメント（READ、FIND、HISTOGRAM、GET、UPDATE、DELETE、END TRANSACTION、BACKOUT TRANSACTION）

注意: Natural Optimizer Compiler で提供されるオプションを、最適化されるステートメントを指定するために使用することはできません。「**Optimizer**オプション」に説明があります。

6 NOCSTAT コマンド

Natural Optimizer Compiler で最適化されたプログラムの場合、カタログ時に特定のステートメントをマシンコードに直接変換することができます。 その結果、最適化されたオブジェクトをランタイム時に Natural で実行すると、パフォーマンスを大幅に向上させることができます。

NOCSTAT コマンドでは、カタログされたプログラミングオブジェクトが分析され、統計情報が示されます。これは、プログラムステートメントに NOC による最適化の利点があるかどうか、ある場合はどの程度最適化できているかを判断する場合に役立ちます。

プログラムがカタログされる場合（STOW、CATALL）、Natural コンパイラによって、ソースプログラム内のステートメントに基づいて内部（中間）オブジェクトコードが生成されます。 ほとんどの場合、1 つのソースステートメントは 1 つの中間コード命令に変換されます。 ただし、FOR や REPEAT などの複雑なステートメントの場合、複数の中間コード命令が生成されます。 NOCSTAT 分析は、生成された中間コード命令に基づきます。 そのため、統計レポートで示されるステートメントの数は、ソースプログラム内のステートメントの数を超える場合があります。

NOCSTAT の呼び出し

手順 6.1. NOCSTAT コマンドを呼び出すには

- ダイレクトコマンド「NOCSTAT」を入力します。

  NOCSTAT のメイン画面が表示されます。

```
14:02:01            ***** NATURAL NOCSTAT COMMAND *****            2000-09-04



Name ................. ________

Library ............... SAGTEST_




NOCable Objects only .. _


Output Report ......... X Statement Category
                        _ Statement Type
                        _ Code Profile
```

```
Output Destination .... X Screen

                        _ CSV to Work File 1

                        _ XML to Work File 1

                          with XSL ____________________________________


Progress Control ...... X




Command ===>

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                  Canc
```

フィールド固有のヘルプ情報を表示するには、関連するフィールドに疑問符を入力して ENTER キーを押すか、カーソルを目的のフィールドに置いて PF1 キーを押します。 NOCSTAT を終了するには、PF3 キーを押します。

レポートの生成

単一のプログラムまたは一連のプログラムの統計レポートを生成できます。 複数のプログラムを一度に分析する場合、レポートは連続して生成されます。 1 つのレポートを見終わったら、ENTER キーを押して次のレポートを表示します。

NOCSTAT のメインメニューには、以下のオプションがあります。

フィールド	説明	
Name	名前または名前の範囲を入力して、調査するプログラムを指定します。	
	value は、1 つ以上の文字の任意の組み合わせです。	
	value	単一のプログラム。
	*	すべてのプログラム。
	*value**	名前が *value* で始まるすべてのプログラム。
	value>	名前が *value* 以上であるすべてのプログラム。

フィールド	説明	
	value<	名前が *value*. 以下であるすべてのプログラム。
Library	ライブラリの名前を入力するか、範囲を指定します。上記の［Name］フィールドの説明と同じ内容が適用されます。 現在のライブラリがデフォルトです。	
NOCable Objects only	このオプションをマークすると、すでに NOC でコンパイルされているプログラムは除外されます。 マークしないと、NOCSTAT コマンドでは、［Name］および［Library］フィールドで指定したすべての Natural プログラムがデフォルトで選択されます。NOC でコンパイルされているプログラムも含まれます。	
Output Report	任意のオプションをマークして、カテゴリ、タイプ、またはコードプロファイルでステートメントを選択します。 以下の「*ステートメントカテゴリ*」、「*ステートメントタイプ*」、および「*コードプロファイル*」を参照してください。	
Output Destination	以下のオプションのいずれかをマークして、出力フォーマットおよび出力先を指定します。	
	Screen	オンライン表示。
	CSV to Work File 1	コンマ区切りの値のスプレッドシートを生成します。ファイル拡張子 .csv を使用して、その後の処理用にワークファイルを PC に直接書き込みます。 Entire Connection がインストールされている場合は、レポートを PC のみにルーティングできます。

	XML to Work File 1	XML ドキュメントを生成します。ファイル拡張子「.xml」を使用して、その後の処理用にワークファイルを PC に直接書き込みます。
		
		フィールド［with XSL］に値を入力した場合は、XML 出力ドキュメントの先頭に処理命令が追加されます。
		
		`<?xml-stylesheet type="text/xsl" href="`
		`value "?>`
		
		入力する *value* は、スタイルシートの絶対 URL または相対 URL である必要があります。次に例を示します。

フィールド	説明	
		nocstat.xsl または http://natural.software-ag.de/nocstat.xsl XSLT 対応ブラウザによる表示またはバッチ XSLT 実行による変換の際に、ドキュメントは処理命令によって、指定したスタイルシートに従って変換されます。この機能の一般的な使用例は、出力 XML の HTML ページへの変換です。 Natural では、2 つの XSLT スタイルシートが、テキストメンバ NOCSTLS1 および NOCSTLS2 として Natural ライブラリ SYSEXUEX の FNAT システムファイルで提供されています。 以下で説明するように、NOCSTLS1 ではレポートタイプ*ステートメントカテゴリ*のフォーマット命令が提供され、NOCSTLS2 ではレポートタイプ*ステートメントタイプ*のフォーマット命令が提供されます。 ファイル拡張子が .xsl のスタイルシートを、XML ワークファイルが保存されている同じディレクトリにダウンロードします。 Entire Connection がインストールされている場合は、レポートを PC のみにルーティングできます。
Progress Control	ワークファイル 1 出力先にのみ適用されます。 このオプションをマークすると、生成されるレポートにリストされるプログラムごとに短いメッセージがオンライン表示されます。	

レポートフォーマット

以下で説明する3つの出力フォーマットから選択して、分析対象ステートメントについて NOCSTAT によって提供される統計を表示できます。NOC ですでに最適化されているプログラムおよび最適化を検討するプログラムについて、異なるレポートレイアウトが生成されます。 以下のサンプルレポートで相違点を示します。 レポート処理を中断して NOCSTAT のメニューに戻るには、PF3 キーを押します。

- ステートメントカテゴリ
- ステートメントタイプ

- コードプロファイル

ステートメントカテゴリ

オプション Statement Category で生成される統計レポートには、ステートメントのさまざまなカテゴリと対応するオカレンス数、およびプログラムがNOCで最適化されているかどうかに応じて、すでに最適化されているか最適化に適しているステートメントの総数がリストされます。

NOC で最適化されているプログラムの例：

```
14:07:17            ***** NATURAL NOCSTAT COMMAND *****            2000-09-04
               Library SAGTEST  Name NOCTEST1 Type Program

     MCG Options: (ON,OVFLW,INDX,MIX,IO)

      Database Loop:        0
    Database Simple:        0
    SORT / WORK I/O:       29
       FOR / REPEAT:        0
   Screen / Printer:       59
String Manipulation:        6
    Arith / Logical:        0
      Program Calls:        3
   Control Transfer:       49
        Block Start:       25
    Set Environment:        2
   System Functions:        0
      Miscellaneous:        0

   Total Statements:      949
      NOC optimized:      762  ( Ratio:  80 % )
    Longest NOC Run:      180  Statements
```

NOC で最適化されていないプログラムの例：

```
14:13:01            ***** NATURAL NOCSTAT COMMAND *****            2000-09-04
               Library SAGTEST  Name NOCTEST2 Type Program

                     No NOC      NOCable
                    --------     --------
      Database Loop:        0            0
    Database Simple:        0            0
    SORT / WORK I/O:        0            0
       FOR / REPEAT:        0            5
   Screen / Printer:       57            0
String Manipulation:        4            8
    Arith / Logical:        0          491
      Program Calls:        3            0
```

```
    Control Transfer:       19           69
         Block Start:       15            0
      Set Environment:       0            0
    System Functions:        0            0
       Miscellaneous:        0            0

                  Total Statements:      672
                   NOC optimizable:      573  ( Ratio:  85 % )
                   Longest NOC Run:      192  Statements
```

レポートの列およびフィールド：

列	説明
No NOC	最適化に適していないステートメント。
NOCable	最適化に適しているステートメント。
フィールド	
Database Loop	処理ループを生成するデータベースステートメントの数。FIND、READ など。
Database Simple	処理ループを生成しないデータベースステートメント。STORE、UPDATE、DELETE、GET など。
SORT / WORK I/O	SORT およびワークファイルステートメント。
FOR / REPEAT	ループを生成するステートメント。
Screen / Printer	画面およびプリンタ I/O。WRITE、DISPLAY、INPUT など。
String Manipulation	文字列ステートメント。EXAMINE、COMPRESS など。
Arith / Logical	算術および論理ステートメント。MOVE、COMPUTE、IF など。
Program Calls	サブルーチンまたはサブプログラムへの制御の転送。PERFORM、CALLNAT、FETCH など。
Control Transfer	プログラム内のジャンプ。ESCAPE BOTTOM、FOR、REPEAT ループなど。
Block Start	コードブロックを区別する非実行ステートメント。DEFINE SUBROUTINE、AT END など。これらのステートメントは実行されないため、最適化されません。
Set Environment	環境を設定するステートメント。SET CONTROL、SET GLOBALS、SET KEY など。
System Functions	TOTAL、SUM、COUNT、MAX、MIN、*COUNT などのステートメント。
Miscellaneous	最適化には関連せず、したがって NOC では無視される中間コードステートメント。
合計	
Total Statements	プログラムで見つかったステートメントの総数。 上記の NOCSTAT コマンドの概要で説明したように、この数は実際のソースステートメントとは一致しない場合があります。
NOC optimized	最適化されているプログラムの場合、上記の NOCSTAT コマンドの概要で説明したように、これらは NOC でマシンコードに最適化された実際の中間コードステートメントです。
NOC optimizable	最適化されていないプログラムの場合、これは最適化される可能性があるステートメントの数です。 いくつかの要因は NOCSTAT プログラムで考慮されないた

列	説明
	め、この数字は実際の数よりも若干多い場合があります。 例えば、5 つ以上の配列がある SUBSTRING ステートメントは最適化されませんが、「最適化可能」として示されます。
Ratio	Total Statements と NOC optimized ステートメントまたは Total Statements と NOC optimizable ステートメント間の関係（パーセント）。
Longest NOC Run	NOC で最適化されているプログラム： 連続する最適化されているステートメントの数 - フラグメントシーケンスが少ないほど、パフォーマンスは向上します。
	最適化されていないプログラム： プログラムが最適化された場合に予期される連続するステートメントの数。

ステートメントタイプ

オプション「Statement Type」で生成される統計レポートには、単一のステートメントと対応するオカレンス数、および最適化されたオブジェクトに対して生成されたNOCコーディングがリストされます。

NOC で最適化されているプログラムの例：

```
09:21:45             ***** NATURAL NOCSTAT COMMAND *****             2000-09-06
                 Library SAGTEST  Name NOCTEST1 Type Program

        Statement             Number
 ------------------------   --------
 DB AT CONDITION                   6
 READ/WRITE WORK FILE             29
 EXAMINE                           6
 WRITE                            51
 INPUT                             3
 NEWPAGE                           2
 REINPUT                           3
 FIND                              1
 READ                              2
 NOC CODE                        760
 BLOCK START                      18
 ON ERROR                          1
 END                               1
 STOP                              2
 RETURN                            3
 RETURN INLINE                    15
 ESCAPE ROUTINE                    3
 ESCAPE ROUTINE IMMEDIATE          1
 MORE
```

NOC で最適化されていないプログラムの例：

```
09:23:15             ***** NATURAL NOCSTAT COMMAND *****             2000-09-06
                  Library SAGTEST  Name NOCTEST2 Type Program

        Statement             No NOC   NOCable
 ------------------------   --------  --------
 DB AT CONDITION                    6         0
 MOVE/COMPUTE/ASSIGN                0       371
 EXAMINE                            4         0
 COMPRESS                           0         7
 WRITE                             47         0
 INPUT                              2         0
 NEWPAGE                            2         0
 REINPUT                            6         0
 FIND                               1         0
 READ                               1         0
 HISTOGRAM                          1         0
 ELSE/CLOSE LOOP                    0        55
 LOOPEND FOR/REPEAT                 0         5
 BLOCK START                        8         0
 ON ERROR                           1         0
 END                                1         0
 STOP                               2         0
 RETURN                             2         0
 MORE
```

コードプロファイル

オプション「Code Profile」で生成される統計レポートには、最適化に適したソースプログラム内のカテゴリでグループ化されたステートメントの連続するシーケンスが表示されるか、最適化されたプログラムに対して生成された NOC コーディングがリストされます。 オカレンスは強調表示されます。

NOC で最適化されているプログラムの例：

```
09:59:04             ***** NATURAL NOCSTAT COMMAND *****             2000-09-06
                  Library SAGTEST  Name NOCTEST1 Type Program

 Line   Statement
 -----  -------------------------
 0000   ON ERROR
 0000   MCG OPTIONS
 0045   MCG OPTIONS
 0050   NOC CODE
 0050   NOC CODE
 0050   NOC CODE
 0050   NOC CODE
```

```
 1110   SET KEY
 1140   NOC CODE
 1140   NOC CODE
 1145   NOC CODE
 1145   NOC CODE
 1150   NOC CODE
 1150   NOC CODE
 1155   NOC CODE
 1155   NOC CODE
 1160   NOC CODE
 1160   NOC CODE
 MORE
```

NOC で最適化されていないプログラムの例：

```
10:01:36             ***** NATURAL NOCSTAT COMMAND *****             2000-09-06
                Library SAGTEST  Name NOCTEST2 Type Program

 Line   Statement
 -----  -------------------------
 0000   ON ERROR
 0000   MCG OPTIONS
 0100   MOVE/COMPUTE/ASSIGN       <-- NOCable
 0100   MOVE/COMPUTE/ASSIGN       <-- NOCable
 0100   MOVE/COMPUTE/ASSIGN       <-- NOCable
 1920   MOVE/COMPUTE/ASSIGN       <-- NOCable
 1920   FOR                       <-- NOCable
 1920   MOVE/COMPUTE/ASSIGN       <-- NOCable
 1920   FOR/REPEAT IF             <-- NOCable
 1930   COMPRESS                  <-- NOCable
 1940   LOOPEND FOR/REPEAT        <-- NOCable
 1960   MOVE/COMPUTE/ASSIGN       <-- NOCable
 1960   MOVE/COMPUTE/ASSIGN       <-- NOCable
 1970   MOVE/COMPUTE/ASSIGN       <-- NOCable
 1970   MOVE/COMPUTE/ASSIGN       <-- NOCable
 1980   MOVE/COMPUTE/ASSIGN       <-- NOCable
 1980   MOVE/COMPUTE/ASSIGN       <-- NOCable
 1990   MOVE/COMPUTE/ASSIGN       <-- NOCable
 MORE
```

バッチ実行

バッチモードで NOCSTAT レポートを処理するジョブの例を以下に示します。 ジョブの実行後、標準的な転送ツールを使用して、生成されたワークファイルをさらに処理するためにホストから PC に転送できます。

z/OS のサンプルジョブ：

```
//NOCBATCH JOB  (NOC,,,30),CLASS=K,MSGCLASS=X                          00000100
//NATEX EXEC PGM=NATBAT31,REGION=6200K,PARM=('IM=D')                   00000200
//STEPLIB  DD DISP=SHR,DSN=TESTNAT.LOAD                                00000300
//CMPRINT  DD SYSOUT=X                                                 00000400
//CMWKF01  DD DSN='NOC.NOCSTAT.OUT',DISP=(NEW,CATLG),                  00000500
            SPACE=(CYL,(1,1)),UNIT=SYSDA,VOL=SER=SAG001                00000600
//SYSOUT   DD SYSOUT=X                                                 00000700
//CMSYNIN  DD *                                                        00000800
NOCSTAT                                                              00000900
*,library,X,,,X                                                       00001000
.                                                                    00001100
FIN                                                                  00001200
/*                                                                   00001300
```

z/VSE のサンプルジョブ：

```
* $$ JOB JNM=NOCTST,CLASS=5,DISP=D
* $$ LST CLASS=Q,DISP=D
// JOB NOCTST
// ASSGN SYS001,DISK,VOL=xxxxxx,SHR
// DLBL CMWKF01,'NOCSTAT.FILE.ONE',0
// EXTENT SYS001,xxxxxx,1,0,1,150
// EXEC NAT234BA,SIZE=NAT314BA,PARM='SYSRDR'
IM=D,OBJIN=R
/*
ADARUN DBID=185
/*
NOCSTAT
*;library;X; ; ; ;X;
.
FIN
/*
/&
```

BS2000/OSD のサンプルジョブ：

```
/.BAT234 LOGON  NAT,1
/        SYSFILE SYSOUT=NAT314.OUT
/        SYSFILE SYSLST=NAT314.LST
/SKIP   .NOP000
=======================================================================
      NAME   : E.NAT314          S T A R T  B A T C H   N A T U R A L
=======================================================================
/.NOP000 REMARK
/        OPTION  DUMP=YES,MSG=FL
/        FILE    NOCSTAT.OUT,LINK=W01
/        FILE    ADAUSER  ,LINK=DDCARD
/        FILE    $SAG.ADA623.MOD       ,LINK=BLSLIB00
/        SYSFILE TASKLIB=MOD234
/        SYSFILE SYSDTA=(SYSCMD)
/        FILE    NAT314.CMPRMIN,LINK=CMPRMIN
/        DCLJV   NATJV1,LINK=*NATB2JV
/        FILE    $NAT.ADALNK.PARMS,LINK=DDLNKPAR
/        REMARK  %%%%%%%%%%  BATCH-PHASE   %%%%%%%%%%%%%%
/        EXEC    NAT314
NOCSTAT
*,ADE,X, , , ,X, , ,X
.
FIN
```

7 マシンコードのサイズの表示

Natural システムコマンド LIST を使用すると、プログラムがマシンコードにコンパイルされているかどうか、およびマシンコードのサイズを表示できます。

手順 7.1. コンパイルされているプログラムをリストするには

- Natural システムコマンドを入力します。

```
LIST DIR object-name
```

指定したオブジェクトのディレクトリ情報が表示されます。画面の一番下に、マシンコードのサイズ、コンパイルに使用された OPT パラメータ、およびプログラムのカタログに使用された NOC バージョンが表示されます。

LIST コマンドの詳細については、『Natural システムコマンドリファレンス』ドキュメントを参照してください。

8 Optimizer の使用例

以下の例では、どのような場合に Natural Optimizer Compiler が最も効果的に使用され、その機能が示されるかを説明します。

例 1 - 向上なし

以下のプログラムの場合、Natural Optimizer Compiler の使用によって得られるものはありません。

```
DEFINE DATA LOCAL
  1 EMPLOYEES VIEW OF EMPLOYEES
    2 JOB-TITLE
    2 BIRTH
    2 NAME
  END-DEFINE
  FIND EMPLOYEES WITH JOB-TITLE = 'PROGRAMMER' OR = 'ANALYST'
                              OR = 'PROGRAMMER/ANALYST'
                              OR = 'SYSTEM ANALYST'
    DISPLAY JOB-TITLE BIRTH NAME
  END-FIND
  END
```

例 2 - 大幅な向上

以下のプログラムを Natural Optimizer Compiler でコンパイルすると、パフォーマンスは約 30 %向上します。つまり、CPU 負荷が 30 %減少します。 このプログラムでは、IT 従業員の年齢の統計分析が実行されます。 最適化されたステートメントは、太字で示されています。

この例では、NOC による 952 バイトの追加マシンコードによって、オブジェクトサイズは 20.5 %増加しています。

プロファイルパラメータの設定	バッファプールのサイズ	NOC によって生成されたマシンコードのサイズ
OPT=NODBG	5768	952
OPT=OFF	4784	0

```
 DEFINE DATA
 LOCAL
 1 EMPLOY VIEW OF EMPLOYEES
   2 JOB-TITLE     (A25)
   2 BIRTH         (D)
 1 I               (I1)  INIT <1>
 1 CDATE           (D)
 1 NUMB            (N4)
 1 SUMM            (P7.2)
 1 SQUARE          (F8)
 1 DEVI            (F8)
 1 DEVIATION       (N3.4)
 1 MEAN            (P2.3)
 1 AGEDIS          (F8/1:70)
 1 AGEMAX          (F8)
 1 AGEH            (P3)
 1 AGE             (P3)
 1 AGEDAYS         (P15)
 1 LINE            (A71/1:20)
 1 REDEFINE LINE
   2 POINTS        (A1/1:20,0:70)
 END-DEFINE
 *
 MOVE *DATX TO CDATE
 *
 FIND EMPLOY WITH JOB-TITLE = 'PROGRAMMER' OR = 'ANALYST'
   OR = 'PROGRAMMER/ANALYST' OR = 'SYSTEM ANALYST'
AGEDAYS:= CDATE - BIRTH
   AGE:=AGEDAYS / 365
   ADD 1 TO AGEDIS(AGE)                  /* DISTRIBUTION
   ADD 1 TO NUMB
   ADD AGE TO SUMM
   COMPUTE SQUARE = SQUARE + AGE * AGE
 END-FIND
 *
 ***********************************************************************
 * COMPUTE ESTIMATES
 ***********************************************************************
 *
 COMPUTE DEVI = NUMB * SQUARE / (SUMM * SUMM) - 1
 COMPUTE DEVIATION = SQRT(DEVI)
 COMPUTE MEAN = SUMM / NUMB
 *
 ***********************************************************************
 * GRAPHIC DISPLAY
 ***********************************************************************
 *
 FOR I 1 70
   IF AGEDIS(I) > AGEMAX MOVE AGEDIS(I) TO AGEMAX
   END-IF
 END-FOR
```

```
FOR I 1 70
  COMPUTE AGEDIS(I) = AGEDIS(I) * 20 / AGEMAX
END-FOR
FOR I 1 70
  COMPUTE AGEH = 21 - AGEDIS(I)
  IF AGEH < 21 MOVE '*' TO POINTS(AGEH:20,I)
  END-IF
END-FOR
*
************************************************************************
* COMPLETE GRAPHIC DISPLAY
************************************************************************
*
MOVE '!' TO POINTS(*,0)
WRITE TITLE LEFT
 AGEMAX(EM=999)  20X  'DISTRIBUTION OF IT-EMPLOYEES BY AGE'
WRITE NOTITLE NOHDR
LINE(*) /
'0--------10--------20--------30--------40--------50--------60--------'
/  'MEAN='
```

例 3 および 4 - CPU の使用

以下のプログラムは、プログラムをコンパイルするときに選択するオプションに応じて、CPU の使用の違いを示しています。 以下の表は、CPU の使用率を秒およびパーセントで示しています。表内の数値は、IBM z/OS 環境でのテスト実行中に測定されました。値は適用されるハードウェアによって異なるため、一般的な参考値としてのみご利用ください。

```
DEFINE DATA LOCAL
1 #I1         (I4) INIT <1>
1 #I2         (I4) INIT <2>
1 #J1         (I4) INIT <3>
1 #J2         (I4) INIT <4>
1 #F          (I4)
1 #ARR1       (N7/10,5)
1 #ARR2       (N5/10,5)
END-DEFINE
*
FOR #F = 1 TO 1000000
  MOVE #ARR1(#I1,#I2) TO #ARR2(#J1,#J2)
END-FOR
*
END
```

オプション	CPU 秒	CPU パーセント
OFF	8.78	100
ON	0.63	7.18
INDX	0.85	9.68
OVFLW	1.71	19.48
INDX,OVFLW	2.00	22.78
INDX,OVFLW,NODBG	1.61	18.34
INDX,OVFLW,NODBG,NOSGNTR	1.61	18.34
NODBG	0.44	5.01
NOSGNTR	0.63	7.18
NODBG,NOSGNTR	0.44	5.01

```
DEFINE DATA LOCAL
1 #I1         (P7) INIT <1>
1 #I2         (P7) INIT <2>
1 #J1         (N7) INIT <3>
1 #J2         (N7) INIT <4>
1 #K1         (I4) INIT <5>
1 #K2         (I4) INIT <6>
1 #F          (I4)
1 #FIELD1     (P5)
1 #FIELD2     (N5)
1 #FIELD3     (I2)
END-DEFINE
*
FOR #F = 1 TO 500000
*
  #FIELD1:= #I1 - #I2 + (13 * 10 / 5)
  #FIELD2:= #J1 - #J2 + (13 * 10 / 5)
  #FIELD3:= #K1 - #K2 + (13 * 10 / 5)
*
END-FOR
*
END
```

オプション	CPU 秒	CPU パーセント
OFF	18.61	100.00
ON	4.95	26.60
INDX	4.95	26.60
OVFLW	5.38	28.91
INDX,OVFLW	5.38	28.91
INDX,OVFLW,NODBG	5.26	28.26
INDX,OVFLW,NODBG,NOSGNTR	5.09	27.35

オプション	CPU 秒	CPU パーセント
NODBG	4.79	25.74
NOSGNTR	4.81	25.85
NODBG,NOSGNTR	4.63	24.88
NODBG,NOSGNTR,ZD=OFF	4.51	24.23
NODBG,NOSGNTR,ZD=OFF,SIGNCHCK=OFF	4.41	23.70

9 Optimizer Compiler のアクティブ化

Natural Optimizer Compiler をアクティブ化するには、以降のセクションで説明する方法のいずれかを使用します。最初の選択肢が最もスタティック、最後の選択肢が最もダイナミックです。

すべての選択肢で、「*Optimizer オプション*」セクションで説明されている Optimizer オプションが使用されます。 これらのオプションを使用すると、マシンコードを生成する方法とタイミング、使用するトレースオプション、および対象となるアーキテクチャを制御できます。 Optimizer オプションは、Natural Optimizer Compiler の唯一の制御メカニズムです。

マクロ NTOPT

Natural パラメータモジュールのマクロ `NTOPT` を使用すると、リンクされる Natural ニュークリアスに対して Natural Optimizer Compiler をスタティックにアクティブ化できます。 この Natural ニュークリアスが起動されるたびに、同じ Optimizer オプションが再度使用されます。

例 **1**：

```
NTOPT 'INDX,OVFLW,ZD=OFF'
```

例 **2**：

```
NTOPT 'INDX,OVFLW,ZD=OFF,TRGPT',                                        -
      'TRSTMT,OPTLEV03'
```

列 72 にある継続文字 「-」 に注意してください。

使用されているオプション設定については、「**Optimizer オプション**」セクションを参照してください。

ダイナミックプロファイルパラメータ OPT

Natural セッションの開始時に、Natural プロファイルパラメータ `OPT` を指定して、Optimizer Compiler をダイナミックにアクティブ化できます。 `OPT` のシノニムとして、`MCG` を使用できます。 パラメータモジュールの指定は上書きされます。 オプションは、現在のセッションでのみ有効です。

例：

```
OPT=(INDX,OVFLW,ZD=OFF)
```

または

```
MCG=(INDX,OVFLW,ZD=OFF)
```

使用されているオプション設定については、「*Optimizer オプション*」セクションを参照してください。

システムコマンド NOCOPT

Natural セッションを開始すると、Natural システムコマンド `NOCOPT` を使用して Optimizer コマンド画面を呼び出すことができます。 この画面では、Natural の起動時に指定された Natural Optimizer Compiler オプションの現在の設定がモニタリングされます。 設定をオンラインで変更できます。

更新したパラメータ設定は、現在のセッションでのみ有効です。

Natural ステートメント OPTIONS

Natural コンパイラステートメント `OPTIONS` の MCG パラメータでは、プログラム内の個々のステートメントに異なるオプションを設定できるので、マシンコード生成を柔軟かつ強力に制御できます。 したがって、1 つの Natural プログラム内で、NOC を複数回アクティブ化および非アクティブ化して、ステートメントの範囲を異なるオプション設定で囲むことができます。

例

```
OPTIONS MCG=(OVFLW,INDX,ZD=OFF)
```

または

```
OPTIONS MCG=OVFLW,INDX,ZD=OFF
```

MCG パラメータのオプションの文字列を、プラス（+）記号またはマイナス（-）記号で開始することができます。これは、指定しないオプションの値は変更されないまま残り、指定したオプションのみが設定（+）またはリセット（-）されることを意味します。次に例を示します。

例：

```
   OPTIONS MCG=+PGEN          /* turns tracing on

  (statements to be traced)

 OPTIONS MCG=-PGEN          /* turns tracing off
```

文字列を「+」または「-」以外で開始すると、文字列が解析される前にすべてのオプションはリセットされます。

注意: Natural ステートメント OPTIONS では、MCG 以外の Natural コンパイラパラメータも提供されます。

使用されているオプション設定については、「*Optimizer オプション*」セクションを参照してください。

10 Optimizer オプション

Natural Optimizerがアクティブ化されると、このセクションで説明するオプションを設定して、チェックを指定できます。

最適化されるステートメントを指定するためにオプションを使用することはできません。

オプションのリスト

以下の表は、NOCオプションのリストおよび説明です。デフォルト値には下線が引かれています。これは、オプションが存在しない場合に想定される値です。

1つのNOCオプションはカッコまたは一重引用符で囲まれた文字列で構成され（Natural OPTIONS ステートメント内を除く）、複数のオプションはコンマで区切られます。 一部のオプションには値がありますが、一部のオプションはオプション文字列内に存在するだけで環境を変更できます。

以下のルールが適用されます。

- オプションの節は角カッコ [ ] で囲まれています。
- 選択肢は波カッコ { } で囲まれています。
- 各選択肢は縦棒 「|」 で区切られています。
- これらの選択肢から 1 つのみ指定できます。

  ON は Y（Yes）と同じです。

  OFF は N（No）と同じです。
- （該当する場合に）オプションの節 ON または OFF、または等価の値を使用しないで指定されたオプションは、ON に設定されたと解釈されます。 例えば、OVFLW は OVFLW=ON と同じです。
- オプション OFF を除き、指定されたオプションによって最適化は（ON が指定されたのと同じように）オンに切り替わり、デフォルト値が適用されます。 例えば、INDEX は ON,INDEX と同じです。

オプション	説明
ABEND	コンパイル中に Natural Optimizer Compiler が ABEND オプションを検出したときに、Natural Optimizer Compiler によって Natural を直ちに異常終了させるコードが生成されるようにします。 このオプションは単独で使用する必要があり、そうでない場合は無視されます。 他のパラメータはこのオプションによって変更またはリセットされません。 このオプションはデバッグに役立ちます。
CACHE[={ON| OFF |Y|N}]	変数のキャッシュをオンまたはオフに切り替えます。「パフォーマンスの考慮事項」セクションの「変数のキャッシュ」も参照してください。
CPU= /370	対象となるアーキテクチャを指定します。 /370 オプションは IBM および Siemens に有効です。

オプション	説明
DIGTCHCK[={ON|OFF|Y|N}]	同じ型および精度の別の変数に移動するときに、パック型およびアンパック型の数値フィールド（フォーマット P および N）の桁をチェックするかどうかを指定します。 例えば、DIGTCHCK が ON で、アンパック型の数値変数（フォーマット N）に X'FA' などの無効な桁が含まれている場合、同じ精度の別のアンパック型の数値変数に移動すると SOC7（または NAT0954）エラーが発生します。 DIGTCHCK が OFF の場合は、エラーは発生しませんが、生成されるコードは高速になります。
ERRDUMP[={ON|OFF|Y|N}]	コンパイルフェーズでエラー条件が検出された場合に NOC がアベンドするかどうかを指定します。 これは、Natural Optimizer Compiler 自体のデバッグに役立ちます。
INDEX[={ON|OFF|Y|N}]	最適化されたコードで配列の添字について範囲外の値がチェックされるかどうかを指定します。 下記の「警告」も参照してください。
INDX[={ON|OFF|Y|N}]	最適化されたコードで配列の添字について範囲外の値がチェックされるかどうかを指定します。 さらに、RANGE が設定されます。 したがって、このオプションは INDEX=ON,RANGE=ON と同じです。 下記の「警告」も参照してください。
IO[={ON|OFF|Y|N}]	予約済み（将来的に使用される予定）です。
LOOPS[={ON|OFF|Y|N}]	互換性保持の目的でのみ提供されています。 有効ではありません。
MIX[={ON|OFF|Y|N}]	互換性保持の目的でのみ提供されています。 有効ではありません。
NODBG[={ON|OFF|Y|N}]	NODBG=OFF/N（デフォルト）を設定していない場合、最適化されたコードをデバッグするために Natural デバッガを使用できません。これは、最適化されたステートメントの制御を Natural デバッガが受け取らないためです。 追加コードは生成されず、プログラムの実行は遅くならず、CPU 時間がさらに使用されることもありません。 NODBG=ON/Y を設定していない場合、追加コードが生成され、TEST モードがオンに設定されていて Natural デバッガの機能が制限されないかどうかがチェックされます。 「*パフォーマンスの考慮事項*」セクションの「NODBG」も参照してください。
NOSGNTR[={ON|OFF|Y|N}]	パック型数値のみに適用されます。 NOSGNTR=OFF（デフォルト）の場合、算術演算の結果または割り当ての対象である正のパック型数値の記号は、COMPOPT パラメータ PSIGNF に従って設定されます。 NOSGNTR=ON の場合、生成された機械命令の実行結果の記号はそのまま変更されません。「他の **Natural** パラメータの影響」セクションも参照してください。
ON	最適化をオンに切り替えます。 追加のオプションを指定しない場合、各オプションに定義されているデフォルト値が有効です。 下記の「警告」で示すように、このために意図しない結果が発生する場合があります。特に、オプション INDEX、INDX、OVFLW、および RANGE に該当します。
OFF	最適化をオフに切り替えます。

オプション	説明
OPTLEV={ 2|3}	最適化レベルを指定します。プログラムを通過するパスの数とほぼ同じです。 OPTLEV=3 は PGEN が指定されている場合に役立ちます。一部のブランチターゲットは最初のパスでは決定できず、最後のパスで PGEN 出力が行われるためです。そのため、一部の値は誤って表示される場合があります。
OVFLW[={ON| OFF|Y|N}]	算術演算または割り当てでのオーバーフローのチェックが、最適化されたコードに含まれるかどうかを指定します。 下記の「*警告*」も参照してください。
PGEN[={ON| OFF|Y|N}]	最適化されたコードの逆アセンブリが出力されるかどうかを指定します。 このオプションによって、他のすべてのトレースオプションも有効になります。 下記の「***PGEN オプション***」も参照してください。
RANGE[={ON| OFF|Y|N}]	配列の操作で範囲チェックが実行されるかどうかを指定します。 これにより、配列の範囲は、すべてのオペランドの対応する次元の要素数と同じになります。 下記の「*警告*」も参照してください。
SIGNCHCK[={ ON|OFF|Y|N}]	パック型またはアンパック型数値乗算器での乗算の結果で、負のゼロがチェックされるかどうかを指定します。ゼロに負の数を乗算した場合、MP 機械命令によって負のゼロの結果が生成されます。SIGNCHCK がオンの場合、この負のゼロは正のゼロに変換されます。 負のゼロのチェックは、パック型またはアンパック型数値乗算器でのすべての乗算に対して実行されます。
TRENTRY	Software AG での内部使用のためのみ。 このパラメータの設定は変更しないでください。
ZD[={ ON|OFF|Y|N}]	除数のゼロがチェックされるかどうかを指定します。 このオプションを指定した場合、コードが挿入され、プログラムは Natural の ZD プロファイルパラメータに従って動作します。つまり、Natural エラー NAT1302 が発行されるか、結果はゼロです。 このオプションを指定しない場合、除数がゼロの場合は Natural エラー NAT0954 が発生します。 Natural の『*パラメータリファレンス*』ドキュメントの「*ZD - ゼロ割り算のチェック*」も参照してください。

注意: INDEX、INDX、OVFLW、および RANGE について：
値 OFF および N の適用に注意してください。 オーバーフローのチェックまたは配列の添字のチェックを抑制すると、不正なプログラムによって、予期しない結果、ストレージ破損、異常終了が発生する場合があります。
下記の「***INDEX および OVFLW の例***」も参照してください。INDEX および OVFLW の影響について説明しています。

- INDEX および OVFLW の例

- 最適なコード生成

INDEX および OVFLW の例

```
DEFINE DATA LOCAL
...
1 P1 (P1/9)
...
1 P3 (P3/9)
...
1 I (I4)
1 J (I4)
1 K (I4)
1 L (I4)
END-DEFINE
...
P1(I:J) := P3(K:L)
...
END
```

例の説明

INDX=ON または INDEX=ON が設定されると、I、J、K、および L が P1 および P3 それぞれに対して定義された範囲内にあることを確認するコードが生成されます。

INDX=ON または RANGE=ON が設定されると、I:J および K:L が同じ長さの範囲を示すことを確認するコードが生成されます。

OVFLW=ON が設定されると、P3 の値が対応する P1 変数に適合することを確認するコードが生成されます。

例：ここでは、値 100 でオーバーフローが発生します。

エラー状況の例：

P3 のオカレンスの 1 つに値 100 が含まれており、OVFLW=OFF が設定されている場合、対応する P1 のオカレンスに割り当てられる値はゼロになります。 添字変数 I がゼロまたは 9 よりも大きく、INDX=OFF が設定されている場合、配列 P1 に属していないストレージエリアが破損します。これらのオプション（OVFLW および INDX）が ON に設定されている場合は、標準の Natural ランタイムでの場合と同じように Natural エラーが発生します。

上記で指定されている NOC オプションでは、追加コードが生成されます。 ただし、これはデバッグが困難なエラーを防ぐなど、チェックによってもたらされる利点によって十分に相殺されます。 検出されないエラーによって予期しない結果が発生することがあります。

最適なコード生成

コード生成が最小になり、最適なパフォーマンスが得られるように、以下の設定を使用します。

```
OPT='NODBG,NOSGNTR,SIGNCHCK=OFF,ZD=OFF'
```

ただし、この設定は、完全にデバッグされたプログラミングオブジェクトに対してのみ使用してください。「*警告*」も参照してください。

PGEN オプション

PGEN オプションを指定すると、Natural Optimizer Compiler によって、生成されたコードおよび内部 Natural 構造が出力されます。 したがって、バグ修正、パフォーマンスの確認、サポートの問題などのために、コードおよび構造を調べることができます。

PGEN オプションで生成された結果を解釈するには、IBM の /370 アセンブラを理解している必要があります。

このオプションは、担当地域の Software AG のサポートを受けて使用することをお勧めします。

- PGEN の設定
- PGEN オプションのサブオプション
- PGEN オプションの出力
- PGEN 出力の操作

PGEN の設定

PGEN 機能を使用するには、Optimizer Compiler をアクティブ化するときに PGEN オプションを設定します。

バッファはメモリ内に保持されるため、ユーザースレッドがトレース情報を保持するのに十分な大きさにならない可能性があります。 この場合は、例えば、プログラムのトレースが必要な部分に対してのみ PGEN をオンに設定してみます。

OPTIONS MCG=(PGEN=ON,TRGPT=ON) または OPTIONS MCG=+PGEN,TRGPT	トレースをオンにします。GPT エントリのトレースが含まれます。
OPTIONS MCG=(PGEN=OFF) または OPTIONS MCG=-PGEN	トレースをオフにします。

さまざまなオプションが出力の内容に影響します。 基本的な PGEN オプションによって、Natural ソース行および対応するコードの逆アセンブリのフォーマットされたリストが生成され、NOCSHOW

ユーティリティによる抽出のためにメモリ内に保持されます。下記の「*PGEN オプションの出力*」で説明します。

TRSTMT、TRGPT、TRMPT、および TRVDT オプションによって、各行に関連付けられた内部データ構造の 16 進ダンプが出力されます。

TRBASES および TRCACHE オプションによって、ベースレジスタおよびキャッシュ変数に関する情報が出力されます。

PGEN オプションのサブオプション

以下の表では、PGEN=ON の場合のオプションについて説明します。 使用される構文については、上記の「*オプションのリスト*」の説明を参照してください。

オプション	説明
LPP={5|..| 55 |..|255}	トレース出力のページごとの行数。TREXT=ON の場合にのみ使用されます。
NOsrcE[={ON| OFF |Y|N}]	NOsrcE=OFF の場合、Natural ソースステートメントが出力に含まれます。
TRACELEV={ 0 |..|255}	トレースレベルを指定します。 この 1 バイト値の各ビットによって、トレースするバッファタイプが指定されます。これらのビットは、TR*xxx* オプションを使用して設定することもできます。
TRBASES[={ON| OFF |Y|N}]	ベースレジスタ割り当てがトレースされるかどうかを指定します。
TRCACHE[={ON| OFF|Y|N}]	CACHE エントリがトレースされるかどうかを指定します。
TREXT[={ON| OFF |Y|N}]	TREXT=ON の場合、以下で説明するように、トレースはユーザー出口 NOCPRINT に送られます。
TRGPT[={ON| OFF |Y|N}]	GPT エントリがトレースされるかどうかを指定します。
TRMPT[=ON| OFF |Y|N}]	MPT エントリがトレースされるかどうかを指定します。
TRSTMT[={ON| OFF |Y|N}]	STMT エントリがトレースされるかどうかを指定します。
TRVDT[={ON| OFF |Y|N}]	VDT エントリがトレースされるかどうかを指定します。

以下の例も参照してください。

PGEN オプションの出力

Natural Optimizer Compiler では、PGEN の出力を以下の 2 つの場所に送ることができます。

- 内部バッファ

- ユーザー出口 NOCPRINT

内部バッファ

このバッファの内容は、CHECK、CAT、STOW、または RUN コマンドが実行されるたびに上書きされます。 システムユーティリティ NOCSHOW が用意されており、このバッファの内容を表示、検索、または印刷できます。

手順 10.1. NOCSHOW ユーティリティを呼び出すには

- Natural Optimizer Compiler がアクティブなときに、ダイレクトコマンド NOCSHOW を CHECK、STOW、CAT、または RUN の後に入力します。

以下の PF キーを画面で使用できます。

キー	機能
PF2	出力の先頭に位置付けます。
PF4	1 行前に位置付けます。
PF5	1 行後に位置付けます。
PF6	Natural プリンタサポート No.1 に印刷します。
PF7	1 ページ前に位置付けます。
PF8	1 ページ後に位置付けます。
PF10	テキスト文字列をスキャンします。
PF11	スキャンを繰り返します。

ユーザー出口 NOCPRINT

TREXT=ON が指定された場合、Natural Optimizer Compiler によって、すべての出力行はトレースバッファに追加されるのではなく、ユーザー出口 NOCPRINT に渡されます。

NOCPRINT は、通常の OS レジスタ規則に従って呼び出されます。 レジスタ 1 は、81 バイトの印刷行のアドレスを含むフルワードをポイントします。行の位置 1 に ANSI 改行制御文字があります。 レジスタ 13 は、18*4 バイトのエリアをポイントします。このエリアはセーブエリアとして使用できます。 レジスタ 14 にはリターンアドレスが含まれ、レジスタ 15 には NOCPRINT のエントリアドレスが含まれます。

ユーザー出口 NOCPRINT は、上記のレジスタ規則をサポートする任意の言語で書き込むことができます。 Natural Optimizer Compiler ニュークリアスに加え、Natural ニュークリアスにリンクする必要があります。

PGEN 出力の操作

このセクションでは、PGEN オプションで作成された出力を解釈する方法について説明します。

- PGEN 出力の先頭に、ソース行の一部とは思われない逆アセンブルされた行があります。 これらは、最適化されていないコードから最適化されているコードへ制御が渡るときに常に実行される前処理を構成する命令です。 永続的なベースレジスタがロードされ、前処理の正しいポイントに制御が渡されます。 下記の「*サンプルセクション A*」を参照してください。
- 複数のソース行がコードなしで出力される場合があります。 これは、Natural コンパイラによって、複数行にまたがる場合があるステートメントのオブジェクト内の単一の行番号が出力されるためです。 下記の「*サンプルセクション B*」を参照してください。
- NODBG=OFF（デフォルト）が指定されている場合、各 Natural ステートメントの開始時に一連の命令が生成されます。

```
BALR R9,R11
DC X'....'
```

この一連の命令によって、（エラーがある場合に）行番号が設定され、TEST モードが ON に切り替えられているかどうかがチェックされます。この一連の命令がないと、NOC でコンパイルされているステートメントの Natural デバッガによるデバッグはできません。 下記の「*サンプルセクション C*」を参照してください。

- 逆アセンブルされた行の間に改行がある場合があります。 この改行は、内部ステートメントの区切りを示しています。 これは、単一の Natural ステートメントによって複数の内部（中間コード）ステートメントが生成される場合があるために発生します。

サンプルセクション **A**：

```
000000 5880 D354           L     R8,RTADR+4
000004 5870 D370           L     R7,RTADR+32
000008 4810 6006           LH    R1,6(,R6)
00000C 1F60                SLR   R6,R0
00000E 47F1 A000           BC    15,0(R1,R10)
```

サンプルセクション **B**：

```
0010 OPTIONS MCG=(PGEN=ON,TRGPT=ON)
0020 DEFINE DATA LOCAL
0030 1 I(I4)
0040 1 P(P7.2)
0050 1 T(P7.2)
0060 END-DEFINE
0070 *

0080 SETTIME
```

```
    0090 *

   000012 45E0 B040           BAL   R14,RETH
   000016 0036                DC    X'0036'

    0100 FOR I=1 TO 100000
```

サンプルセクション **C**：

```
000018 059B               BALR  R9,R11
   00001A 003E               DC    X'003E'
   00001C D203 7000 833B     MVC   I,#VAR033B

   000022 059B               BALR  R9,R11
   000024 004C               DC    X'004C'
   000026 47F0 A040          BC    15,64(,R10)

   00002A 059B               BALR  R9,R11
   00002C 005A               DC    X'005A'
   00002E BFFF 8343          ICM   R15,15,#VAR0343
   000032 BF0F 7000          ICM   R0,15,I
   000036 1A0F               AR    R0,R15
   000038 BE0F 7000          STCM  R0,15,I

   00003C 059B               BALR  R9,R11
   00003E 006C               DC    X'006C'
   000040 BFFF 833F          ICM   R15,15,#VAR033F
   000044 BF0F 7000          ICM   R0,15,I
   000048 190F               CR    R0,R15
   00004A 4720 A066          BC    2,102(,R10)

    0110   ADD 1.00 TO P

   00004E 059B               BALR  R9,R11
   000050 0082               DC    X'0082'
   000052 FA41 7004 8347     AP    P,#VAR0347
   000058 DC00 7008 B488     TR    P+4(1),PSGNTR

    0120 END-FOR
    0130 *

   00005E 059B               BALR  R9,R11
   000060 0094               DC    X'0094'
   000062 47F0 A02A          BC    15,42(,R10)

    0140 T:=*TIMD(0080)

   000066 059B               BALR  R9,R11
   000068 009C               DC    X'009C'
   00006A 45E0 B0D8          BAL   R14,SYSFUNC
   00006E 0330 B881          DC    X'0330B881'
```

```
000072 F246 7009 8330      PACK  T,#VAR0330
000078 F040 7009 0002      SRP   T,2,0
00007E DC00 700D B488      TR    T+4(1),PSGNTR

 0150 T:=T / 10
 0160 *

000084 059B                BALR  R9,R11
000086 00AE                DC    X'00AE'
000088 F864 D100 7009      ZAP   OP1(7),T
00008E F811 D130 8349      ZAP   WORK2(2),#VAR0349
000094 45E0 B104           BAL   R14,ZDCHECK
000098 F240 7009 B355      PACK  T,ZEROZ
00009E 47F0 E01C           BC    15,28(,R14)
0000A2 FD61 D100 8349      DP    OP1(7),#VAR0349
0000A8 D204 7009 D100      MVC   T,OP1
0000AE DC00 700D B488      TR    T+4(1),PSGNTR

 0170 DISPLAY 'ELAPSED TIME (S)' T

0000B4 45E0 B040           BAL   R14,RETH
0000B8 00C0                DC    X'00C0'

 0180 END
```

他の Natural パラメータの影響

グローバルパラメータ ZD は、NOC コンパイラの動作に影響します。 上記の「*オプションのリスト*」で説明されている ZD オプションの説明を参照してください。

COMPOPT パラメータ PSIGNF（Naturalの『システムコマンド』ドキュメントのシステムコマンド COMPOPT も参照）は、正のパック 10 進数の符号を ON の場合は F、OFF の場合は C にすることによって、動作に影響します。 このパラメータは、NOSGNTR=OFF が指定された場合に適用されます。

パック型データ（フォーマット P）については、以下の図を参照してください。

NOSGNTR=OFF	および	PSIGNF=ON	すべての符号は F に正規化されます（デフォルト）。
NOSGNTR=OFF	および	PSIGNF=OFF	すべての符号は C に正規化されます。
NOSGNTR=ON			すべての符号は最後の操作で生成されたままです。

数値データ（フォーマット N）の場合、NOSGNTR および PSIGNF の設定に関係なく、符号は常に F に正規化されます。

11 パフォーマンスの考慮事項

フォーマット

算術演算でパック型数値（P）および整数（I4）のデータフォーマットを使用する場合に、最適なパフォーマンスが得られます。

パック型数値（P）、アンパック型数値（N）、整数（I）、および浮動小数点（F）のフォーマット間のデータ変換は避けます。最適化されたコードであっても、処理オーバーヘッドが発生するためです。

最適化されたコードでは、解釈のオーバーヘッドがないため、さまざまなデータフォーマット間の相違はより顕著になります。例えば、最適化されたコードの場合にフォーマット N ではなく P を使用することで得られるパフォーマンスの向上は、通常のコードの場合よりも大きくなります。

例：

```
A = A + 1
```

上記の数値計算の場合

- 最適化されていないコードでは、フォーマット P での実行はフォーマット N よりも約 13 %速くなります。
- ただし、最適化されたコードでは、フォーマット P での実行はフォーマット N よりも約 56 %速くなります。

この単純なステートメントに Natural Optimizer Compiler を適用することで得られるパフォーマンスの向上は、以下のとおりです。

- アンパック型オペランド（N）の場合：8 倍高速
- パック型オペランド（P）の場合：15 倍高速

配列

以下のような配列範囲処理は、

```
MOVE A(*) TO B(*)
```

同じ機能を FOR ステートメント処理ループを使用してプログラムした場合よりも、効率的に実行されます。 このことは、最適化されたコードにも該当します。

索引を使用する場合、最適なパフォーマンスを得るには整数フォーマット I4 を使用する必要があります。

英数字フィールド

英数字定数を英数字変数（フォーマットA）に移動する場合、または英数字変数を英数字定数と比較する場合、英数字定数の長さを変数の長さに合わせて調整することをお勧めします。 これによって、処理は大幅に速くなります。例えば、以下のようにすると

```
A(A5):='XYZAB'


...
IF A = 'ABC  ' THEN ...
```

以下の場合よりも速くなります。

```
IF A = 'ABC' THEN ...
```

DECIDE ON

`DECIDE ON` ステートメントをシステム変数、配列、またはパラメータ *operand1* とともに使用する場合、`LOCAL` ストレージセクションで定義されている同じタイプおよび長さのスカラ変数に値を移動したほうが効率的です。

数値

割り当てまたは算術演算で数値定数を使用する場合は、定数が処理と同じタイプになるようにしてください。

一般的なルール

- 小数の有無は任意だが指数のない数値定数は、値を表現する最小の長さと精度を持つパック型数値にコンパイルされます。ただし、定数が配列の添字またはサブストリングの開始位置または長さの場合は、4バイトの整数（I4）になります。このルールは、演算に参加する変数のタイプに関係なく適用されます。

- 浮動小数点を含む演算は、浮動小数点で実行されます。 EOO を数値に追加して、強制的に浮動小数点にします。以下に例を示します。

```
ADD 1E00 to F(F8)
```

- 浮動小数点は含まないが、パック型数値、アンパック型数値、日付または時刻変数を含む演算は、パック10進で実行されます。 ADD、SUBTRACT、および IF の場合、小数位と末尾のゼロを追加して、数値定数が一番精度の高い変数と同じ数の小数位を持つようにします。以下に例を示します。

```
ADD 1.00 TO P(P7.2)
```

この方法は MULTIPLY および DIVIDE には必要ありません。

変数の位置付け

最適化処理を簡単にするには、すべてのスカラ参照をデータセクションの最初に、すべての配列参照をデータセクションの最後に置くようにします。

変数のキャッシュ

Natural Optimizer Compiler には、パフォーマンスをさらに高めるアルゴリズムが含まれています。 パフォーマンスの観点では、ステートメントはオペランドのタイプによって異なります。1つ以上のオペランドがパラメータ、配列、またはタイプN（数値）のスカラフィールド、あるいはこれらのオペランドの組み合わせである場合、ステートメントの実行は遅くなります。NOC では、プログラムフローが分析され、これらの特性の1つ以上を持つどの変数が、書き込みなしで複数回読み込まれるかが判別されます。 次に、各変数の値は、以下の条件に従って、一時キャッシュエリアに移動されてすばやくアクセスされます。

- 変数は、複数回アクセスされるが、ほとんど変更されない。かつ
- 変数は、任意のタイプの配列またはタイプ N（数値）のスカラフィールドである。

変数のキャッシュに最も適しているのは、同じ変数に繰り返しアクセスする長いシーケンスを含むプログラムであり、特に変数が配列である場合です。 変数のキャッシュによって、複雑で何度も繰り返されるアドレス計算が回避されます。

変数のキャッシュの例

以下のサンプルプログラムは、変数のキャッシュの利点を示しています。NODBG（下記参照）および CACHE=ON でカタログして、このプログラムをテスト環境で実行すると、NODBG および CACHE=OFF でプログラムを実行する場合に必要となる時間の 47 %かかりました。 プログラムを CACHE=ON でカタログすると、NOC によって生成されるコードは 856 バイトから 376 バイトに削減されます。

```
DEFINE DATA LOCAL
1 ARR(N2/10,10,10)
1 I(I4) INIT <5>
1 J(I4) INIT <6>
1 K(I4) INIT <7>
END-DEFINE
DECIDE ON EVERY ARR(I,J,K)
 VALUE 10 IGNORE
 VALUE 20 IGNORE
 VALUE 30 IGNORE
 VALUE 40 IGNORE
 VALUE 50 IGNORE
 VALUE 60 IGNORE
 VALUE 70 IGNORE
 VALUE 80 IGNORE
 VALUE 90 IGNORE
 NONE IGNORE
END-DECIDE
```

注意: キャッシュされている変数の内容を Natural デバッガのコマンド MODIFY VARIABLE で変更すると、元の変数の内容のみが変更されます。 キャッシュされている値（後続のステートメントで使用される可能性がある）は変更されません。 そのため、Natural デバッガを使用する場合は、変数のキャッシュの使用に注意する必要があります。 Natural の『デバッガ』ドキュメントも参照してください。

NODBG

プログラムを完全にテストして実稼働に移したら、「Optimizer オプション」セクションで説明されているように、プログラムを **NODBG** オプションでカタログする必要があります。 デバッグコードがないと、最適化されたステートメントの実行は 10 %～30 %速くなります。

このオプションを指定すると、INDX または OVFLW オプションがオンになっていても、デバッグを容易にするためのコードは削除されます。

12 Zap のリスト

サイトで Natural Optimizer Compiler に適用されている Zap の概要を取得する場合は、DUMP システムコマンドを使用します。

手順 12.1. Zap の概要を取得するには

- Natural システムコマンドを入力します。

  ```
  DUMP ZAPS NOC
  ```

  適用されている Zap のリストが表示されます。

Natural Optimizer Compiler に Zap が適用されていない場合は、そのことを示すメッセージが表示されます。