

Intel Corporation
5200 N.E. Elam Young Parkway
Hillsboro, OR 97124-6497

(503) 696-8080



May 1995

Dear Paragon™ System Customer:

This package contains Release 5.0 of the Paragon™ System Fortran compiler for your Paragon system.

Before using your system:

- **Read this letter completely.**
- **Verify the contents of this package.**
- **Read the *Paragon™ System Fortran Compiler Release 5.0 Release Notes*.**

Package Contents

Your Paragon System Fortran compiler software package should include one of the cartridge tapes listed in Table 1 and the documentation listed in Table 2. If any items are missing, or if you have any questions, please contact Intel Scalable Systems Division as described in the "Comments and Assistance" section.

Table 1. Installation Media

Description	Order Number
Paragon™ System Fortran Compiler Release 5.0 Sun4/SunOS-4 and Sun4/SunOS-5 Hosted	633954-001 ✓
Paragon™ System Fortran Compiler Release 5.0 Silicon Graphics Hosted	633955-001



Table 2. Documentation

Description	Order Number
<i>Paragon™ System Fortran Compiler Release 5.0 Release Notes</i>	633953-001 ✓
<i>Paragon™ System Fortran Compiler User's Guide</i>	312491-003 ✓
<i>Paragon™ System Fortran Language Reference Manual</i>	312644-002 ✓
<i>Paragon™ System Fortran System Calls Reference Manual</i>	312488-003 ⁰⁰⁴ ✓
<i>Effective Fortran 77</i>	312201-001

Restrictions and Limitations of Compiler Release 5.0

Every effort has been taken to ensure the quality of this release, but at shipping time we are aware of a few problems. Please refer to the *Paragon™ System Fortran Compiler Release 5.0 Release Notes* for known limitations and workarounds.

Installation

For directions on how to install your Paragon Fortran compiler, refer to the *Paragon™ System Fortran Compiler Release 5.0 Release Notes*.

NOTE

You must have System Software Release 1.3 installed on your system in order to install Compiler Release 5.0.

Comments and Assistance

Intel Scalable Systems Division is eager to hear of your experiences with the Paragon System Fortran compiler. Please call us if you need assistance, have questions, or otherwise want to comment on your Paragon system.

U.S.A./Canada Intel Corporation
Phone: 800-421-2823
Internet: support@ssd.intel.com

Intel Corporation Italia s.p.a.
Milanofiori Palazzo
20090 Assago
Milano
Italy
1678 77203 (toll free)

France Intel Corporation
1 Rue Edison-BP303
78054 St. Quentin-en-Yvelines Cedex
France
0590 8602 (toll free)

Intel Japan K.K.
Scalable Systems Division
5-6 Tokodai, Tsukuba City
Ibaraki-Ken 300-26
Japan
0298-47-8904

United Kingdom Intel Corporation (UK) Ltd.
Scalable Systems Division
Pipers Way
Swindon SN3 IRJ
England
0800 212665 (toll free)
(44) 793 491056
(44) 793 431062
(44) 793 480874
(44) 793 495108

Germany Intel Semiconductor GmbH
Dornacher Strasse 1
85622 Feldkirchen bei Muenchen
Germany
0130 813741 (toll free)

World Headquarters
Intel Corporation
Scalable Systems Division
15201 N.W. Greenbrier Parkway
Beaverton, Oregon 97006
U.S.A.
(503) 677-7600 (Monday through Friday, 8 AM to 5 PM Pacific Time)
Fax: (503) 677-9147

If you have comments about our manuals, please fill out and mail the enclosed Comment Card. You can also send your comments electronically to the following address:

techpubs@ssd.intel.com (Internet)



Intel Supercomputer Users' Group

The Intel Supercomputer Users Group promotes the exchange of information among users. Intel strongly supports the Users Group and encourages participation in its activities, which include: Special Interest Groups (SIGs), an annual international users conference, an electronic mail task force, and a "freeware" library of user-contributed software, available electronically to all members of the Intel Supercomputer Users' Group. For membership information contact:

JoAnne Wold (503-677-5322)
joanne@ssd.intel.com (Internet)

Sincerely,



Peter Wolochow

Product Marketing Manager
Intel Scalable Systems Division

Paragon is a registered trademark of Intel Corporation
Silicon Graphics is a registered trademark of Silicon Graphics, Inc.
Sun Microsystems is a trademark of Sun Microsystems.

Copyright © 1995 Intel Corporation



May 1995

Order Number: 633953-001

Paragon™ System Fortran Compiler
Release 5.0
Release Notes

Intel® Corporation

Copyright ©1995 by Intel Scalable Systems Division, Beaverton, Oregon. All rights reserved. No part of this work may be reproduced or copied in any form or by any means...graphic, electronic, or mechanical including photocopying, taping, or information storage and retrieval systems...without the express written consent of Intel Corporation. The information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update or to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

Intel software products are copyrighted by and shall remain the property of Intel Corporation. Use, duplication, or disclosure is subject to restrictions stated in Intel's software license agreement. Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraphs (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at 252.227-7013. Intel Corporation, 2200 Mission College Boulevard, Santa Clara, CA 95052-8119. For all Federal use or contracts other than DoD, Restricted Rights under FAR 52.227-14, ALT. III shall apply.

The following are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products:

286	i386	Intel	iPSC
287	i387	Intel386	Paragon
i	i486	Intel387	
	i487	Intel486	
	i860	Intel487	

APSO is a service mark of Rational Corporation

DGL is a trademark of Silicon Graphics, Inc.

Ethernet is a registered trademark of XEROX Corporation

EXABYTE is a registered trademark of EXABYTE Corporation

Excelan is a trademark of Excelan Corporation

EXOS is a trademark or equipment designator of Excelan Corporation

FORGE is a trademark of Applied Parallel Research, Inc.

Green Hills Software, C-386, and FORTRAN-386 are trademarks of Green Hills Software, Inc.

GVAS is a trademark of Verdix Corporation

IBM and IBM/VS are registered trademarks of International Business Machines

Lucid and Lucid Common Lisp are trademarks of Lucid, Inc.

NFS is a trademark of Sun Microsystems

OpenGL is a trademark of Silicon Graphics, Inc.

OSF, OSF/1, OSF/Motif, and Motif are trademarks of Open Software Foundation, Inc.

PGI and PGF77 are trademarks of The Portland Group, Inc.

PostScript is a trademark of Adobe Systems Incorporated

ParaSoft is a trademark of ParaSoft Corporation

SCO and OPEN DESKTOP are registered trademarks of The Santa Cruz Operation, Inc.

Seagate, Seagate Technology, and the Seagate logo are registered trademarks of Seagate Technology, Inc.

SGI and SiliconGraphics are registered trademarks of Silicon Graphics, Inc.

Sun Microsystems and the combination of Sun and a numeric suffix are trademarks of Sun Microsystems

The X Window System is a trademark of Massachusetts Institute of Technology

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.

VADS and Verdix are registered trademarks of Verdix Corporation

VAST2 is a registered trademark of Pacific-Sierra Research Corporation

VMS and VAX are trademarks of Digital Equipment Corporation

VP/ix is a trademark of INTERACTIVE Systems Corporation and Phoenix Technologies, Ltd.

XENIX is a trademark of Microsoft Corporation

WARNING

Some of the circuitry inside this system operates at hazardous energy and electric shock voltage levels. To avoid the risk of personal injury due to contact with an energy hazard, or risk of electric shock, do not enter any portion of this system unless it is intended to be accessible without the use of a tool. The areas that are considered accessible are the outer enclosure and the area just inside the front door when all of the front panels are installed, and the front of the diagnostic station. There are no user serviceable areas inside the system. Refer any need for such access only to technical personnel that have been qualified by Intel Corporation.

CAUTION

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

LIMITED RIGHTS

The information contained in this document is copyrighted by and shall remain the property of Intel Corporation. Use, duplication or disclosure by the U.S. Government is subject to Limited Rights as set forth in subparagraphs (a)(15) of the Rights in Technical Data and Computer Software clause at 252.227-7013. Intel Corporation, 2200 Mission College Boulevard, Santa Clara, CA 95052. For all Federal use or contracts other than DoD Limited Rights under FAR 52.2272-14, ALT. III shall apply. Unpublished—rights reserved under the copyright laws of the United States.

Preface

These release notes provide the latest information on Release 5.0 of the Paragon™ system Fortran compiler.

These release notes assume that you are an application programmer proficient in the Fortran language and the UNIX operating system.

Organization

- | | |
|-----------|--|
| Chapter 1 | Introduces the new features of the Release 5.0 Fortran compiler. |
| Chapter 2 | Contains installation instructions for native and cross-compilers. |
| Chapter 3 | Describes guidelines and limitations for this release. |
| Chapter 4 | Contains open and fixed bug lists. |

Notational Conventions

This manual uses the following notational conventions:

- | | |
|---------------|---|
| Bold | Identifies command names and switches, system call names, reserved words, and other items that must be used exactly as shown. |
| <i>Italic</i> | Identifies variables, filenames, directories, processes, user names, and writer annotations in examples. Italic type style is also occasionally used to emphasize a word or phrase. |

Plain-Monospace

Identifies computer output (prompts and messages), examples, and values of variables. Some examples contain annotations that describe specific parts of the example. These annotations (which are not part of the example code or session) appear in *italic* type style and flush with the right margin.

Bold-Italic-Monospace

Identifies user input (what you enter in response to some prompt).

Bold-Monospace

Identifies the names of keyboard keys (which are also enclosed in angle brackets). A dash indicates that the key preceding the dash is to be held down *while* the key following the dash is pressed. For example:

<Break> <s> <Ctrl-Alt-Del>

- [] (Brackets) Surround optional items.
- ... (Ellipsis dots) Indicate that the preceding item may be repeated.
- | (Bar) Separates two or more items of which you may select only one.
- { } (Braces) Surround two or more items of which you must select one.

Applicable Documents

For more information, refer to the *Paragon™ System Technical Documentation Guide*.

Comments and Assistance

Intel Scalable Systems Division is eager to hear of your experiences with our products. Please call us if you need assistance, have questions, or otherwise want to comment on your Paragon system.

U.S.A./Canada Intel Corporation
Phone: 800-421-2823
Internet: support@ssd.intel.com

Intel Corporation Italia s.p.a.
Milanofiori Palazzo
20090 Assago
Milano
Italy
1678 77203 (toll free)

France Intel Corporation
1 Rue Edison-BP303
78054 St. Quentin-en-Yvelines Cedex
France
0590 8602 (toll free)

Intel Japan K.K.
Scalable Systems Division
5-6 Tokodai, Tsukuba City
Ibaraki-Ken 300-26
Japan
0298-47-8904

United Kingdom Intel Corporation (UK) Ltd.
Scalable Systems Division
Pipers Way
Swindon SN3 IRJ
England
0800 212665 (toll free)
(44) 793 491056 (*answered in French*)
(44) 793 431062 (*answered in Italian*)
(44) 793 480874 (*answered in German*)
(44) 793 495108 (*answered in English*)

Germany Intel Semiconductor GmbH
Dornacher Strasse 1
85622 Feldkirchen bei Muenchen
Germany
0130 813741 (toll free)

World Headquarters
Intel Corporation
Scalable Systems Division
15201 N.W. Greenbrier Parkway
Beaverton, Oregon 97006
U.S.A.
(503) 677-7600 (Monday through Friday, 8 AM to 5 PM Pacific Time)
Fax: (503) 677-9147

If you have comments about our manuals, please fill out and mail the enclosed Comment Card. You can also send your comments electronically to the following address:

techpubs@ssd.intel.com
(Internet)

Table of Contents

Chapter 1 Product Features

Release 5.0 Features	1-1
Making Loops Parallel (-Mconcur)	1-1
Making Loops with Calls Parallel (-Mncall)	1-2
Getting Information About Parallel Loops	1-3
Directives to Support Parallel Loops	1-3
altcode (count) concur	1-3
altcode (count) concurreduction	1-3
noaltcode	1-3
(no)cncall	1-3
(no)concur	1-3
dist=block	1-4
dist=cyclic	1-4
Other New Switches	1-4
Default Compiler Switch Settings	1-5

Chapter 2

Installation

Installing the Native Compiler	2-1
Installing the Cross-Development Compilers	2-5

Chapter 3

Guidelines and Limitations

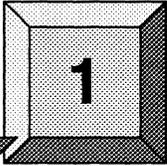
Guidelines for Using the Fortran Compiler	3-1
PostScript Copies of the Manuals and Release Notes	3-2

Chapter 4

Bug Lists for Fortran Compilers

Introduction	4-1
Open Bug List	4-2
Fixed Bug List	4-3

Product Features



1

Release 5.0 Features

Several new compiler switches and directives have been added to support the parallel execution of loops.

The compiler is able to use the three separate processors of an MP node by making some loops parallel by splitting execution of the loop among two or three processors. Each processor is allocated certain iterations of the loop to perform. This can result in greater performance. Both inner and outer loops can be parallelized. For nested loops, the compiler selects the outermost valid loop and makes it parallel.

A loop can be parallelized if its iterations can be performed in any order without affecting the results computed by the loop. For example, one type of loop that cannot be parallelized is one in which the results of some iteration are used in a later iteration. Loops with reductions, such as vector sum or dot product, fit this description. The compiler will try to parallelize this type of loop, but can only do so by performing the sums in a different order than defined by the original loop. As a result, the final sum computed may be slightly off due to roundoff error. If exact results are important, you can use the **-Mconcur=noassoc** switch to prevent parallelization of loops with reductions.

The following sections describe the compiler switches and directives associated with parallelizing loops.

Making Loops Parallel (-Mconcur)

The **-Mconcur** switch causes the compiler to parallelize certain loops. The following options are available:

-Mconcur=altcode:count

Make innermost loops without reduction parallel only if their iteration count exceeds *count*. Without this switch, the compiler assumes a default *count* of 100.

-Mconcur=altcode_reduction:count	Make innermost loops with reduction parallel only if their iteration count exceeds <i>count</i> . Without this switch, the compiler assumes a default <i>count</i> of 200.
-Mconcur=dist:block	Make the outermost valid loop parallel. This is the default option.
-Mconcur=dist:cyclic	Make the outermost valid loop in any loop nest parallel. If an innermost loop is made parallel, its iterations are allocated to processors cyclically. That is, processor 0 performs iterations 0, 3, 6, ...; processor 1 performs iterations 1, 4, 7, ...; and processor 2 performs iterations 2, 5, 8, and so on.
-Mconcur=global_vcache	Directs the vectorizer to locate the cache within the area of an external array when generating codes for parallel loops. By default, the cache is located on the stack for parallel loops.
-Mconcur=noassoc	Do not make loops with reductions parallel.

Making Loops with Calls Parallel (-Mncall)

By default, the compiler does not parallelize loops with calls, since there is no way for the compiler to verify that the called routines are safe to execute in parallel. The **-Mncall** switch forces the compiler to parallelize loops with calls. When you specify **-Mncall** on the command line, the compiler also automatically specifies **-Mreentrant**.

-Mncall also allows several other types of loops to be made parallel:

- loops with I/O statements
- loops with conditional statements
- loops with low loop counts
- non-vectorizable loops

If the compiler can detect a cross-iteration dependency in a loop, it will not make the loop parallel, even if **-Mncall** is specified.

Getting Information About Parallel Loops

In addition to providing information about vectorization, the **-Minfo=loop** switch now also provides information about any loop parallelization that has occurred.

The **-Mneginfo=concur** switch prints information for each countable loop that is not made parallel stating why the loop was not made parallel.

Directives to Support Parallel Loops

The following directives have been added to support the parallel loop features of the compiler.

altcode (*count*) concur

This directive sets the loop count threshold for parallelization of non-reduction loops to *count*. The default loop count is 100. Under this directive, innermost loops without reductions are executed in parallel only if their iteration count exceeds *count*.

altcode (*count*) concurreduction

This directive sets the loop count threshold for parallelization of loops with reduction to *count*. The default loop count is 200. Under this directive, innermost loops with reductions are executed in parallel only if their iteration count exceeds *count*.

noaltcode

This directive sets the loop count threshold for parallelization of all innermost loops to 0.

(no)cncall

This directive alters the effects of the **-Mncall** command line switch. The **cncall** directive causes the compiler to consider loops within the specified scope for parallelization, even if they contain calls to user-defined routines, they contain conditional statements, their loop counts do not exceed the usual thresholds, or they contain inner non-vectorizable loops. If you use the **cncall** directive, you must specify **-Mconcur** on the compiler command line.

(no)concur

This directive alters the effects of the **-Mconcur** command line switch. The **concur** directive causes the compiler to consider loops within the specified scope for parallelization. If you use the **concur** directive, you must specify **-Mconcur** on the compiler command line.

dist=block

This directive changes the concurrency characteristics to block within the scope of the directive.

dist=cyclic

This directive changes the concurrency characteristics to cyclic within the scope of the directive.

Other New Switches

-Mclr_reg	Clear the internal registers after every procedure invocation. This option is used for diagnostic purposes.
-Mcpp860	Direct the internal preprocessor to not compress white space.
-Mcray	Enable Cray compatibility mode for various options.
-Mnoansi	Allow multiple implicit statements.
-Mretain_static	Do not eliminate static data that is not referenced.
-Munroll[=<i>option</i> [,<i>option</i> ...]]	Invoke the loop unroller and set the optimization level to 2 if it is set to less than 2. <i>option</i> is one of the following: c:m - Completely unroll loops with a constant loop count less than or equal to <i>m</i> . If <i>m</i> is not supplied, the default value is 4. n:u - Unroll loops that are not completely unrolled or have a non-constant loop count <i>u</i> times. If <i>u</i> is not supplied, the unroller computes the number of times a loop is unrolled.
-Mnounroll	Do not unroll loops.
-Mvect=altcode[:<i>number</i>]	Produce non-vectorized code to be executed if the loop count is less than or equal to <i>number</i> . Otherwise execute vectorized code. The default value for <i>number</i> is 10.

For a complete description of these switches, see the *Paragon™ System Fortran Compiler User's Guide*

Default Compiler Switch Settings

The default compiler switch settings are set for ease of porting, safe optimization, and high-speed compilation. Some of the defaults are:

- O1** Optimization level one
- Mnostride0** Do not check for zero stride induction variables.
- Mnodebug** Debugging disabled
- Mperfmon** Performance monitoring enabled
- Mnoframe** Don't include stack frame pointers on stack
- Kieee** Math conforms to IEEE 754 standard
- Mdepchk** Assume that potential data dependencies exist
- Msplit_loop_ops=40**
Split innermost loops whose number of floating-point operations exceeds 40 if **-Mvect** is specified.
- Msplit_loop_refs=20**
Split innermost loops whose number of array element loads and stores exceeds 20 if **-Mvect** is specified.

For better performance, you may use values other than the defaults, or change your defaults with a configuration file. For example, some appropriate user-defined defaults might be:

- O2** Optimization level two
- Mnoperfmon** No performance monitoring
- Knoieee** Non-IEEE math, if floating point accuracy is not critical
- Mnodepchk** Assume that no potential data dependencies exist

If you use these suggested values as user-defined defaults, then in order to debug the program you have to override several of them. For example, to debug, you would want to use the **-g** command line switch. The **-g** switch is equivalent to the following:

- O0 -Mframe -Mdebug**

For best performance you may need to override the suggested defaults with command line switches such as the following:

-O3 or **-O4**

-Mvect

For more information on **-Mnostride0**, **-Knoieee**, **-Mvect**, and other switches, see the *Paragon™ System Fortran Compiler User's Guide*

NOTE

If your application contains a loop with an induction variable whose increment (stride) is zero, you should add the **-Mstride0** switch to the compiler command line. **-Mstride0** is no longer the default.

Installing the Native Compiler

Installation Time:	Approximately 45 minutes.
Installation Media:	One 0.25-inch QIC 150 cartridge tape labelled "Paragon™ System Fortran Compiler Release 5.0 Sun4/SunOS-4 and Sun4/SunOS-5 Hosted (633954-001)."

The installation tape, "Paragon™ System Fortran Compiler Release 5.0 Sun4/SunOS-4 and Sun4/SunOS-5 Hosted (633954-001)," contains the following files:

<i>nat_ftn.tar.Z</i>	The native Fortran compiler
<i>sun_ftn.tar.Z</i>	The Sun 4 Fortran cross-compiler
<i>sol_ftn.tar.Z</i>	The Solaris Fortran cross-compiler
<i>native_install</i>	Script to install the native compiler
<i>cross_install</i>	Script to install the cross-compilers
<i>if77.doc.tar.Z</i>	Online documentation

NOTE

Install the compilers after installing system software installation.

NOTE

These instructions assume that you are reading the tape on the Paragon diagnostic station. You may be able to read the tape on some other networked system; but if you have difficulty, use the diagnostic station.

1. Log in to the diagnostic station as *root*.
2. Copy the installation **tar** files from the release tape into */u/tmp* on the diagnostic station.
3. First make */u/tmp* your working directory. Then perform the following steps.

DS# **cd /u/tmp**

- A. Insert the release tape into the cartridge tape drive on the diagnostic station.
- B. Issue the command,

✓ DS# **tar xvf /dev/rStp0 native_install nat_ftn.tar.Z
if77.doc.tar.Z**

- C. After the file has been copied, remove the tape from the cartridge tape drive.
4. Log in to the Paragon system as *root*. During the installation, *umask* will be set to 022 by the installation script. It will be restored to the original value before the installation script completes. If the compiler is installed in a directory other than */*, and this directory is created outside the installation script, the ownership and permissions will be set correctly by the installation script.
 5. If you have already installed the native C compilers and */tmp/native_install* still exists on your system, you can proceed to step 6.

Establish an **ftp** connection with the diagnostic station and transfer the following file:

native_install This file copies the compiler and documentation files from the diagnostic station and installs them in */* or an alternate directory.

On the Paragon system, issue the following commands:

```
# cd /tmp ✓
# ftp diagnostic station IP address
ftp> cd /u/tmp
ftp> get native_install ✓
ftp> bye
# chmod 544 native_install
```

6. Execute the installation script.

```
# cd / ✓
# /tmp/native_install f77
```

The following is displayed. The distribution information is read from */etc/defaults/install* if the file exists:

```
=====
Native Compiler Installation
=====

Root directory for compiler installation [path]: /
Temporary storage location on Paragon:          /tmp
Distribution Node:                               myhost ✓
Distribution Path:                               /my_default_path
Is this correct? [y/n]:
```

To change any of these values, answer “n” to the “Is this correct?” prompt and enter the desired value when prompted to do so. If you enter <CR> at a prompt, the value is not changed. If you change the value of the root installation directory, and the directory does not exist, you are asked if you want to create it. When you are satisfied with all the values displayed, enter “y” in response to the “Is this correct” prompt.

The files are copied to the Paragon system and installed. The following is an example of the output seen when installing in the directory /.

```
Username for FTP'ing files from myhost: [anonymous] myname
```

```
.
.
.
FTP output from file transfers
.
.
.
```

```
221 Goodbye.
Uncompressing nat_ftn.tar.Z...
Uncompressing if77.doc.tar.Z...
Installing Native FORTRAN compiler...
```

Native FORTRAN compiler has been installed
Installing FORTRAN manual pages...

Installation complete

7. Verify that your path is set correctly.

If the root directory for the install was not */*, set *PARAGON_XDEV* to be the root directory you entered, and add *\$PARAGON_XDEV/usr/bin* to the beginning of your execution path. You must also add *\$PARAGON_XDEV/usr/man* to the beginning of your *MANPATH* environment variable to access the R5.0 manual pages. If the environment variables *PARAGON_LPATH* or *LPATH* are defined, and they contain directories where R4.5 versions of libraries reside, these variables should be modified to use *\$PARAGON_XDEV/usr/lib* instead.

The following should display when you use the compiler **-VV** switch. If it does not, examine your *PATH* environment variable and make any needed corrections.

✓ **if77 -VV**

```
if77/Paragon Paragon Version R5.0
Copyright 1995, Intel Corporation and The Portland Group Inc.
All Rights Reserved
```

View *\$PARAGON_XDEV/usr/share/release_notes/if77_5.0_release_notes.ps*
for a list of new features for Release R5.0

8. Execute the installation verification test.

```
✓ # cd root_installation_directory/usr/testinstall
  # ./testinstall_f
  Installation successful
```

9. Remove the *testinstall* directory.

```
✓ # cd ..
  # rm -rf testinstall
```

Installing the Cross-Development Compilers

Installation Time:	Approximately 45 minutes.
Installation Media:	One 0.25-inch QIC 150 cartridge tape labelled "Paragon™ System Fortran Compiler Release 5.0 Sun4/SunOS-4 and Sun4/SunOS-5 Hosted (633954-001)."
Installation Media (SGI):	One 0.25-inch QIC 150 cartridge tape labelled "Paragon™ System Fortran Compiler Release 5.0 Silicon Graphics Hosted (633955-001)."

The installation tape, "Paragon™ System Fortran Compiler Release 5.0 Sun4/SunOS-4 and Sun4/SunOS-5 Hosted (633954-001)," contains the following files:

<i>nat_ftn.tar.Z</i>	The native Fortran compiler
<i>sun_ftn.tar.Z</i>	The Sun 4 Fortran cross-compiler
<i>sol_ftn.tar.Z</i>	The Solaris Fortran cross-compiler
<i>native_install</i>	Script to install the native compiler
<i>cross_install</i>	Script to install the cross-compilers
<i>if77.doc.tar.Z</i>	Online documentation

The installation tape, "Paragon™ System Fortran Compiler Release 5.0 Silicon Graphics Hosted (633955-001)," contains the following files:

<i>sgi_ftn.tar.Z</i>	The native Fortran compiler
<i>cross_install</i>	Script to install the cross-compilers
<i>if77.doc.tar.Z</i>	Online documentation

The cross-development tools and compilers are installed by reading in a set of **tar** files from the installation tape onto the diagnostic station. You then **ftp** one or more compressed **tar** files to your workstation or workstation server, where you **untar** them. Do not install the cross-development compilers on the Paragon.

1. Log in to the diagnostic station as *root*.
2. Copy the installation **tar** files from the release tape into */u/tmp* on the diagnostic station. Each compressed **tar** file is about 4M bytes. After installation each compiler requires about 8M bytes. The total space for all compiler-related files can be as much as 78M bytes.

A. First, make */u/tmp* your working directory, Then perform the following steps.

```
DS# cd /u/tmp
```

B. Insert the release tape into the cartridge tape drive on the diagnostic station.

C. Extract the compressed tar files and installation script. The installation script and documentation files are the same for each host and can be extracted only once.

If you are copying the cross-compiler for a Sun4 workstation, issue the command:

```
✓ DS# tar xvf /dev/rStp0 cross_install sun_ftn.tar.Z  
if77.doc.tar.Z
```

If you are copying the cross-compiler for an SGI workstation, issue the command:

```
DS# tar xvf /dev/rStp0 cross_install sgi_ftn.tar.Z  
if77.doc.tar.Z
```

If you are copying the cross-compiler for a Sun4/Solaris workstation, issue the command:

```
✓ DS# tar xvf /dev/rStp0 cross_install sol_ftn.tar.Z  
if77.doc.tar.Z
```

D. After the files have been copied, remove the tape from the cartridge tape drive.

3. If you have installed new system software, you must copy the system libraries and include files to the cross-development environment. If you have not installed new system software, proceed to step 4. This step may take up to 30 minutes to complete.

Log in to the Paragon system as *root*. Then:

```
✓ # cd /tmp  
# /usr/bin/mksysfiles  
The default base directory is set to /.  
Do you wish to change it (y/n)? [n]
```

If you installed the system libraries and header files relative to */* (that is, under */usr/ccs/lib*, */usr/include*), choose *n*. Otherwise, enter *y* to be prompted for a new base directory.

```
✓ # exit
```

4. If you have already installed the C cross compiler(s) and */tmp/cross_install* still exists on your system, you can proceed to step 5.

Establish an **ftp** connection with the diagnostic station and transfer the following file:

cross_install This file copies the compiler(s) and documentation files from the diagnostic station and installs them.

On your workstation, issue the following commands:

```

✓ CROSS# cd /tmp
✓ CROSS# ftp diagnostic station IP address
ftp> cd /u/tmp
✓ ftp> get cross_install
ftp> bye
CROSS# chmod 544 cross_install

```

5. If you do not need to create a new directory for the R5.0 compilers, you can proceed to step 6.

```

✓ CROSS# mkdir directory
CROSS# chmod 755 directory

```

6. Make the directory in which the cross compiler(s) will be installed your current working directory and execute the installation script.

```

CROSS# cd directory
CROSS# /tmp/cross_install f77

```

The following is displayed. Please read the explanatory information following this menu before proceeding.

```

=====
                          Cross Compiler Installation
=====

```

```

Install Sun4? [y/n]:           Y
Install Solaris? [y/n]:       Y
Install SGI? [y/n]:           Y
Root directory for compiler installation [path]: $PARAGON_XDEV
Location for compressed tar files [path]:       /tmp
Distribution host name:        unknown
Distribution host user name for ftp:            anonymous
Distribution Path:             /u/tmp
Install system files? [y/n]:   n
Create links for system files? [y/n]:          n
Is this correct? [y/n]:        n

```

To change any of these values, answer “n” to the “Is this correct?” prompt and enter the desired value when prompted to do so. If you enter <CR> at a prompt, the value is not changed. When you are satisfied with all the values displayed, enter “y” in response to the “Is this correct” prompt.

If you copied the compiler files into a local or NFS-mounted file system, you can avoid using **ftp**. To do this, enter the pathname of the directory containing the compiler files for `Location` for compressed tar files and ***noremove*** for `Distribution` host name.

If you respond with “y” to the “Install system files” prompt, you are asked for the following information:

```
Enter name of paragon system where sysfiles.tar.Z was created:
Enter user name for ftp from your_system: [anonymous]
Enter path of sysfiles.tar.Z on your_system: [/tmp]
```

This will add approximately 45 minutes to the installation and will require approximately 130M bytes during the installation.

If you respond with “y” to the “Create links for system files” prompt, you are asked for the following information:

```
Enter root directory of actual files:
```

All of the compressed **tar** files needed for your installation selections are copied to the location you specified. Each file is uncompressed and installed, and the uncompressed tar files are deleted. If any of the `uncompress` or `tar` commands fail, the installation is aborted. The most likely cause for a failure is lack of disk space. If this occurs, you may need to install one compiler at a time.

In the following example, the user installs the Sun4 compiler, the Sun4/Solaris compiler, and the SGI compiler.

Assume that the compressed **tar** files were previously read from the installation tape(s) into `/u/tmp` on a system named *fred*. The script transfers compressed **tar** files from *fred* to the directory `/vol/scratch/tmp` on the local system via **ftp**, and you are asked for the *root* password on *fred*.

The file *sysfiles.tar.Z* had been previously created in `/tmp` on the Paragon system *my_paragon*. The script then transfers *sysfiles.tar.Z* on *my_paragon* and places it in `/vol/scratch/tmp` on the local system. You are asked for the *root* password for *my_paragon*.

The compilers are installed in `/vol/scratch/install`.

CROSS# */tmp/cross_install f77* ✓

```
=====
                        Cross Compiler Installation
=====
```

```
Install Sun4? [y/n]:           Y
Install Solaris? [y/n]:       Y
Install SGI? [y/n]:           Y
Root directory for compiler installation [path]: /vol/scratch/install
Location for compressed tar files [path]:       /vol/scratch/tmp
Distribution host name:         fred
Distribution host user name for ftp:            root
Distribution path:              /u/tmp
Install system files? [y/n]:   Y
  Paragon system name:         my_paragon
  Paragon user name for ftp:    root
  Path for compressed system tar file: /tmp
Create links for system files? [y/n]:          n
Is this correct? (y/n):         y ✓
```

```
Connected to fred.
220 fred FTP server (SunOS 4.1) ready.
331 Password required for root.
Password:
230 User root logged in.
200 Type set to I.
Local directory now /vol/scratch/tmp
200 PORT command successful.
```

```
.
.
.
  ftp files from distribution system
.
.
.
```

```
221 Goodbye.
Connected to my_paragon.
220 my_paragon FTP server (OSF/1 Version 5.60) ready.
331 Password required for root.
Password:
```

```
.
.
.
  ftp file from Paragon system
.
.
```

```

221 Goodbye.
Uncompressing sysfiles.tar.Z...
Installing system libraries and header files
Uncompressing if77.doc.tar.Z...
Installing FORTRAN manual pages...
Uncompressing sun_ftn.tar.Z...
Installing Sun FORTRAN compiler...
Uncompressing sol_ftn.tar.Z...
Installing Sun4/Solaris FORTRAN compiler...
Uncompressing sgi_ftn.tar.Z...
Installing SGI FORTRAN compiler...

```

✓ Installation complete

CROSS#

7. Verify that your path is set correctly.

Set *PARAGON_XDEV* to be the root directory you entered, and add *\$PARAGON_XDEV/paragon/bin."arch"* to the beginning of your execution path. For example, on Sun4/Solaris systems you would add *\$PARAGON_XDEV/paragon/bin.solaris*. You must also add *\$PARAGON_XDEV/paragon/man* to the beginning of your *MANPATH* environment variable to access the R5.0 manual pages. If the environment variables *PARAGON_LPATH* or *LPATH* are defined, and they contain directories where R4.5 versions of libraries reside, these variables should be modified to use *\$PARAGON_XDEV/paragon/lib-coff* instead.

The following should display when you use the compiler **-VV** switch. If it does not, examine your *PATH* environment variable and make any needed corrections.

✓ CROSS# **if77 -VV**

```

if77/Paragon "host" Version R5.0
Copyright 1995, Intel Corporation and The Portland Group Inc.
All Rights Reserved

```

View *\$PARAGON_XDEV/paragon/release_notes/if77_5.0_release_notes.ps* for a list of new features for Release R5.0

8. Execute the installation verification test. The *testinstall_f* script requires the name of the paragon system where the test will be executed as an argument. You must be able to execute **rcp** and **rsh** commands on the Paragon system you specify.

```

✓ CROSS# cd $PARAGON_XDEV/paragon/testinstall
CROSS# ./testinstall_f paragon_system
Installation successful

```

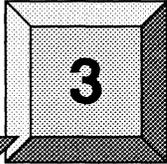
9. Remove the *testinstall* directory.

```
CROSS# cd ..
```

```
CROSS# rm -rf testinstall
```

)

Guidelines and Limitations



3

This chapter describes limitations to the Fortran compiler for this release and provides some guidelines for using the compiler. At the end of this chapter is a current list of bugs for the compiler and a list of the bugs fixed since Release 4.5. The list of bugs is updated just before shipment, and the lists are also available online in the files */usr/share/release_notes/lf77_buglist* and */usr/share/release_notes/lf77_fixed* on the Paragon system.

Guidelines for Using the Fortran Compiler

This section provides some hints and suggestions for making the best use of the compiler.

1. The compiler may occasionally generate internal compiler messages. If they are of severity **W** (Warning) or **I** (Informational), the generated code is correct. However, please report all internal messages to SSD.
2. When using pipelining (**-O4**), the **-Mnodepchk** switch generally increases pipelining opportunities. If the program does not produce correct results with this switch, then it must be omitted. Use the switch only if you are sure no data dependencies that inhibit vectorization exist.
3. To ensure that the values of local variables are retained for the next invocation of a subprogram, use the **-Msave** switch. This allows older Fortran programs to run, but can greatly reduce performance.
4. The internal value of `.TRUE.` is `-1` unless the **-Munixlogical** switch is used. In that case, it has a value of `1`.
5. The **-Mvect** switch or the **-O4** switch may generate incorrect results if elements within a vector are accessed with stride zero. If this happens, add the **-Mstride0** switch.

6. The **-Mvect=unroll** switch is no longer supported or documented. The compiler prints the following warning message if you use it.

```
if77 - warning -- Mvect=unroll not implemented
```

7. No features are currently enabled by the **-Mbeta** switch.
8. The **-Mstride0** compiler switch should be used if a loop may contain an induction variable whose increment (stride) is zero. For example:

```
IS = 0
J = 0
DO I = 1,N
  A(J) = B(I) + 1.0
  J = J + IS
ENDDO
```

This switch may degrade performance so should only be used if zero-stride induction variables are possible.

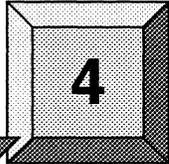
9. The **-Knoieee** switch can give a substantial performance improvement. Division that does not conform to IEEE is several times faster than IEEE division, and some benchmarks run about twice as fast overall with the **-Knoieee** switch set. The penalty you pay for this performance is up to three low order bits of accuracy on certain division operations, and denormals are flushed to zero. The majority of division operations give identical results, whether or not IEEE math is used.
10. If your application runs slower when you use **-Mvect -O4**, try **-Mvect=streamlim:999 -O4**. The additional overhead of streaming in and streaming out data to and from cache could result in decreased performance if the vectors are short.
11. For applications with array references that are not stride 1, you may see increased performance if you use **-Mvect=streamlim:999**.

PostScript Copies of the Manuals and Release Notes

PostScript copies of the Paragon manuals are available in the directory */usr/share/ps.docs* on the Paragon system. This directory also contains the file *README.if77*, which lists the Fortran compiler manuals contained in the directory.

Postscript copies of the release notes are available in the directory */usr/share/release_notes*.

Bug Lists for Fortran Compilers

4

Introduction

This chapter contains a list of open bugs and a list of fixed bugs. These lists are updated just before shipment and are also available online in the files */usr/share/release_notes/if77_buglist* and */usr/share/release_notes/if77_fixed* on the Paragon system.

The open bug list lists the open bugs against the current release of the Fortran compilers. The bug list includes the following:

- Bug number
- Subsystem name (IF77)
- Bug synopsis
- Bug description

The fixed bug list lists the bugs fixed since the last release of the Fortran com[compilers. The fixed bug list is organized in numerical order by bug number. The bug listing includes the following:

- Bug number
- Subsystem name
- Bug synopsis

These bug lists were generated on 3/15/95.

Open Bug List

The following lists the open bugs for Release 5.0 of the Fortran compilers:

6965 IF77

Synopsis: FORTRAN formatted I/O functions do not work with PFS I/O modes.

Using the FORTRAN formatted I/O functions (such as write and read) in conjunction with PFS I/O modes can lead to program exits and system hangs. For example, using a combination of setiomode() and a FORTRAN write statement can cause a system hang.

There is no workaround for this problem.

11070 IF77

Synopsis: The overhead associated with -Mconcur on compile line is too high.

When a program is compiled and run with one compute processor, the run time can be as much as 10% greater than when the same program is compiled and run with -Mconcur removed from the compile line.

11844 IF77

Synopsis: Bogus compiler error while writing with character constant edit descriptor.

The following code produces this error:

```

Program iotest
write(6,('testing:'))           ! #0 : does NOT work
write(6,('`testing:'))         ! #1 : works
write(6,('\`testing:\`'))      ! #2 : works
write(6,('A') 'testing:')     ! #3 : works
write(6,1)                     ! #4 : works
1  format("testing:")          ! #4
stop
end

```

Apparently, the fio library, encodefmt.c, ef_putstring() ----- function should check for *p = '\"' as well as *p = '\'.

11973 IF77

Synopsis: `sv_rsqrt` returns unreasonable values when dealing with exceptions.

The vector reciprocal sqrt call (`sv_rsqrt`) appears to compute the sqrt instead of $1/\text{sqrt}$ when it runs into an exception case (division by zero, for example). The problem seems to be with the function `__fvzdiv` in `libvect.a`, which is called from `__fvrsqrt`. Apparently, `__fvzdiv` exits immediately if dividend is 0.0.

12262 IF77

Synopsis: Suspicious string copying in Fortran routine.

12551 IF77

Synopsis: Optimizer mishandles argument references not used by all entry points.

The optimizer (with `-O2`) moves references to arguments that are not listed on all entry points to a function.

Fixed Bug List

The following lists the bugs fixed since the last release of the Fortran compiler:

7086 IF77

Synopsis: Compiling with `-Mvect -O4` gives `f_dest` must not be same as `f_src1` error.

9813 IF77

Synopsis: `asin` function call in loop with `Mvect` option inconsistent.

9916 IF77

Synopsis: code causes f77 child caught signal message

10310 IF77

Synopsis: Fortran library function `__fio_open` has uninitialized pointer

10443 IF77

Synopsis: program runs 10 times slower with -Mvect

10450 IF77

Synopsis: simple user code produces different results at different optimizations

10491 IF77

Synopsis: -Mvect -Mconcur causes internal compiler
make_private_indvars: hlv loop not found

10661 IF77

Synopsis: user code produces wrong results with -Mvect at O4 or O3
optimization

10663 IF77

Synopsis: code gives incorrect results with -Mvect at O3 and O4

10870 IF77

Synopsis: Fortran fio_gopen routine has uninitialized pointer.

10902 IF77

Synopsis: if77 with -Msave and -Mvect produced wrong results from
user code

11411 IF77

Synopsis: Electron Density Fails due to an -Mvect problem

11416 IF77

Synopsis: -Mvect undermines reentrancy for codes like PCCM2

11889 IF77

Synopsis: -Msave not compatible with -Mreentrant

12010 IF77

Synopsis: user code illustrates problem with Mvect and *
declaration