Contents

Figures xi

Tables xiii

About This Guide xv
What This Guide Contains xv
What You Should Know Before Reading This Guide xvi
Suggestions for Further Reading xvi
Conventions Used in This Guide xviii

1. About the MIPSpro Compiler System 1

2. Using the MIPSpro Compiler System 7
Selecting a Compiler 8
Using Command-Line Options 8
Setting an Environment Variable 8
Object File Format and Dynamic Linking 9
Executable and Linking Format 9
Dynamic Shared Objects 10
Position-Independent Code 10
Source File Considerations 11
Source File Naming Conventions 11
Header Files 12
Specifying a Header File 12
Creating a Header File for Multiple Languages 13
Using Precompiled Headers in C and C++ 13
About Precompiled Headers 14
Automatic Precompiled Header Processing 14
Other Ways to Control Precompiled Headers 18
PCH Performance Issues 19

Compiler Drivers 20
Default Behavior for Compiler Drivers 20
General Options for Compiler Drivers 21
Linking 25
Invoking the Linker Manually 26
Linker Syntax 26
Linker Example 28
Linking Assembly Language Programs 28
Linking Libraries 28
Specifying Libraries and DSOs 29
Examples of Linking DSOs 30
Linking to Previously Built Dynamic Shared Objects 30
Linking Multilanguage Programs 31
Finding an Unresolved Symbol With *ld* 33
Debugging 33

Getting Information About Object Files 34
Disassembling Object Files with *dis* 35
dis Syntax 35
dis Options 35
Listing Parts of DWARF Object Files With *dwarfdump* 36
dwarfdump Syntax 36
dwarfdump Options 37
Listing Parts of ELF Object Files and Libraries with *elfdump* 38
elfdump Syntax 38
elfump Options 38
Determining File Type with *file* 40
file Syntax 41
file Example 41
Listing Symbol Table Information: *nm* 41
nm Syntax 41
nm Symbol Table Options 42
nm Example of Obtaining a Symbol Table Listing 43
Determining Section Sizes with *size* 45
size Syntax 45
size Options 45
size Example 46
Removing Symbol Table and Relocation Bits with *strip* 46
strip Syntax 46
Using the Archiver to Create Libraries 47
ar Syntax 48
ar Options 49
ar Examples 50

3. Using Dynamic Shared Objects 53
Benefits of Using DSOs 53

Using DSOs 55
DSOs vs. Archive Libraries 55
Using QuickStart 56
Guidelines for Using Shared Libraries 56
Choosing Library Members 57
Tuning Shared Library Code 58
Taking Advantage of QuickStart 60
Building DSOs 63
Creating DSOs 63
Making DSOs Self-Contained 63
Controlling Symbols to Be Exported or Loaded 64
Using DSOs With C++ 65
Using Registry Files 66
Registry File Format 67
Directive Lines 67
Shared Object Specification Lines 68
Run-Time Linking 69
Searching for DSOs at Run Time 69
Run-Time Symbol Resolution 70
Compiling with **–Bsymbolic** 70
Converting Libraries to DSOs 72
Dynamic Loading Under Program Control 73
Versioning of DSOs 75
The Versioning Mechanism 75
What Is a Version? 76
Building a Shared Library Using Versioning 76
Example of Versioning 77

4. Using the Performance Tools 81
Overview of Profiling 82

Profiling With *prof* 82
Running the Profiler 82
prof Syntax 83
prof Defaults 83
prof Options 84
pc Sampling 86
Obtaining pc Sampling 86
Creating Multiple Profile Data Files 89
pc Sampling Frequency 89
Examples of Using *prof* to Obtain pc Sampling Analysis Data 89
Example Using prof –pcsample 89
Basic Block Counting 91
Using pixie 91
pixie Syntax 91
pixie Options 92
Obtaining Basic Block Counts 93
Examples of Basic Block Counting 96
Example Using prof –pixie –invocations 96
Example Using prof –pixie –heavy 97
Example Using prof –pixie –lines 98
Example Using prof –pixie –quit 99
Example Using prof –pixie –procedures 100
Example Using prof –pixie –procedures –clock 101
Example Using prof –pixie –dis 102
Summing Basic Block Count Results From Multiple Runs 103
Inclusive Basic Block Counting 104
Syntax 105
Example of prof –gprof –pixie 107
Profiling Multiprocessed Executables 110
Rearranging Procedures With *cord* 111
cord Syntax 111
cord Options 112
Example Using *cord* 112

5. Optimizing Program Performance 117
Optimization Overview 117
Benefits of Optimization 117
Optimization and Debugging 118
Using the Optimization Options 118
Optimization Options 118
Controlling Floating Point Optimization 119
–OPT:roundoff=n 120
–OPT:IEEE_arithmetic=n 121
Other Options to Control Floating Point Behavior 122
Debugging Floating-Point Problems 124
Controlling Miscellaneous Optimizations 125
Controlling Loop Unrolling 126
Controlling Global Code Motion 127
Controlling Software Pipelining 129
Controlling the Target Architecture 134
Controlling the Target Environment 135
Improving Global Optimization 137
Overview of the Global Optimizer 137
Optimizing C and Fortran Programs 138
Optimizing C Programs 138
Example of Pointer Placement and Aliasing 139
Improving Other Optimization 141
C and Fortran Programs 141
C Programs Only 142
Register Allocation 143

6. Coding for 64-Bit Programs 147
Coding Assumptions to Avoid 147
sizeof(int) == sizeof(void *) 147
sizeof(int) == sizeof(long) 148
sizeof(long) == 4 148
sizeof(void *) == 4 148
Implicitly Declared Functions 149
Constants With the High-Order Bit Set 149
Arithmetic with **long** Types 149
Solving Porting Problems 150
Guidelines for Writing Code for 64-Bit Silicon Graphics Platforms 150

7. Porting Code to N32 and 64-Bit Silicon Graphics Systems 155
Compatibility 155
N32 Porting Guidelines 157
Porting Environment 158
Source Code Changes 158
Build Procedure 158
Runtime Issues 159
Porting Code to 64-Bit Silicon Graphics Systems 159
Using Data Types 160
Using Predefined Types 161
Using Typedefs 162
Using Large Files With XFS 163
164

Index 165
We'd Like to Hear From You 177
Three Ways to Reach Us 177