

Contents

Contents	i
Tables	x
Figures.....	xi
About This Book	xiii
Summary of Contents	xiii
Document Conventions	xv
Notational and Typographic Conventions	xv
Terminology Conventions	xv
Where to Go for More Information	xvi
Documents in the PowerPC Toolset.....	xvi
C and C++ Programming Documents	xvii
Processor-Specific Documents	xvii
Specifications and ABI Documents.....	xviii
 1	
Introducing High C/C++.....	1
1.1 About the C Language.....	1
1.1.1 High C Extensions to Standard C	1
1.2 About the C++ Language	2
1.3 About the High C/C++ Compiler	2
1.4 Tools and Utilities Used with High C/C++	3
1.5 Standards, Conformance, and Portability	3
 2	
Using the Compiler.....	5
2.1 The Build Process.....	5
2.2 High C/C++ Driver Overview	7
2.2.1 Driver Syntax.....	7
2.2.2 Driver Command	7
2.2.3 Compiler Options	8
2.2.4 Driver Input Files.....	8
2.2.5 Driver Argument Files.....	10
2.2.6 Executable File Names	11
2.3 Using the High C/C++ Driver	11

2.3.1	Getting Help	11
2.3.2	Compiling and Linking in One Step	12
2.3.3	Compiling without Linking	13
2.3.4	Generating Source and Assembly Listings	13
2.3.5	Preparing Your Program for Debugging	14
2.4	Linking Run-Time Libraries	14
2.4.1	Linking MetaWare Libraries and System Libraries	15
2.4.2	Including Additional Libraries	15
2.4.3	Locating Libraries	16
2.5	Customizing the Compilation	16
2.5.1	Setting Compiler Controls	17
2.5.2	Setting Compile-Time Environment Variables	19
2.5.3	Using the Macro Preprocessor	20
2.5.4	Using Profiles	22
2.5.5	Using the Driver Configuration (.cnf) File	24
2.5.6	Using Makefiles	24

3

Using Compiler Options	27
3.1 Specifying Compiler Options	27
3.2 Compiler Option Reference	27
3.3 Format Strings for Output-File Names	54

4

Using Compiler Toggles	57
4.1 Specifying Toggles	57
4.2 Toggle Reference	58

5

Using Compiler Pragmas	95
5.1 Setting Pragmas	95
5.2 Pragma Reference	96
5.3 Customizing Your Program with Pragmas	107
5.3.1 Setting Module Initialization Priority: Pragma Initialization_level	108
5.3.2 Evaluating Expressions at Run Time: Pragmas Startup_ expr and Exit_expr	108
5.3.3 Aliasing External Names: Pragmas Alias and Global_ aliasing_convention	109
5.3.4 Mapping Variables to Control Sections: Pragma Data ...	113

5.3.5	Specifying an Alternate Code Section: <code>Pragma Code</code>	115
5.3.6	Specifying an Alternate Data Section: <code>Pragma Static_</code> <code>segment</code>	116
5.3.7	Specifying an Alternate Literals Section: <code>Pragma</code> <code>Literals</code>	117
5.3.8	Placing Comments in Object Code: <code>Pragma Ident</code>	117

6

Optimizing Program Performance	119
6.1 Overview	119
6.2 Optimization Levels	120
6.2.1 Specifying an Optimization Level	120
6.3 Execution Speed Versus Code Size	121
6.3.1 How Optimizations Can Affect Speed and Size	122
6.4 Debugging Optimized Code	124
6.5 Optimization Reference	125
6.5.1 Aligning Code and Data	125
6.5.2 Back-Substitution of Epilog Code	126
6.5.3 Back-Substitution of Nodes (Jump Reduction Inside Nodes) .	126
6.5.4 Code Assumed “Well Behaved”	126
6.5.5 Code in Source-Code Order	127
6.5.6 Code Space Minimizing (Optimizing for Space)	127
6.5.7 Common Subexpression Elimination (Global)	127
6.5.8 Common Subexpression Elimination (Local)	127
6.5.9 Common Subexpression Elimination (Local, Iterative) ...	128
6.5.10 Compiler Intrinsics	128
6.5.11 Constant Expression Folding	129
6.5.12 Constant Propagation	129
6.5.13 Constants in Code	129
6.5.14 Cross Jumping (Tail Merging)	130
6.5.15 Dead (Unreachable) Code Elimination	130
6.5.16 Expression Simplification	130
6.5.17 Inlining ANSI Standard C Functions	131
6.5.18 Inlining Functions	131
6.5.19 In-Order Execution of I/O (Disable)	133
6.5.20 Instruction Scheduling (High-Level)	134
6.5.21 Instruction Scheduling (Low-Level)	135
6.5.22 Live/Dead Analysis	135

6.5.23	Live/Dead Analysis (Iterative).....	135
6.5.24	Loop Induction-Variable Elimination	136
6.5.25	Loop Memory-Reference Elimination	136
6.5.26	Loop Strength Reduction	136
6.5.27	Loop Unrolling.....	137
6.5.28	No Aliasing	137
6.5.29	Operator Strength Reduction.....	138
6.5.30	Preloading Arguments from Memory	138
6.5.31	Register Allocation (Enhanced)	138
6.5.32	Register Allocation (Global)	139
6.5.33	Register Lifetime Analysis.....	139
6.5.34	Single Static Assignment	139
6.5.35	Small-Data Sections Allocation	139
6.5.36	Spill Analysis (Improved)	140
6.5.37	Spill-Code Clean-Up.....	140
6.5.38	Spill-Code Reduction	140
6.5.39	Tail Recursion	141

7

Language Extensions	143
7.1 Semantics of Near and Far	143
7.1.1 Near and Far Functions	143
7.2 Special Type Qualifiers.....	144
7.2.1 Casting Type Qualifiers	145
7.2.2 Access-Related Type Qualifiers.....	145
7.2.3 Endianness-Related Type Qualifiers.....	146
7.2.4 Function Type Qualifiers	147
7.3 Generating Inline Assembly Code with <code>_ASM</code>	149
7.4 The Enhanced <code>asm</code> Facility	153
7.4.1 <code>_asm</code> Macros	153

8

C++-Specific Issues	157
8.1 Compiling and Linking C and C++ Modules	157
8.2 Using the Built-In Inliner Versus the Stand-Alone Inliner	158
8.2.1 Optional Stand-Alone Function Inliner.....	158
8.2.2 Built-in C++ Inliner	159
8.3 Using the Name Unmangler.....	159
8.3.1 Invoking the Name Unmangler	160

8.3.2	Examples of Name Unmangling.....	161
8.4	Using Exception Handling	162
8.4.1	Generating Exception-Aware Classes	162
8.4.2	Making a Class Exception-Aware with <code>Toggle Exception_aware_class</code>	162
8.4.3	Interacting with Previously Compiled Classes	164
8.5	Generating Run-Time Type Information (RTTI)	165
8.6	Switching to New-Style Type Cast Notation: <code>Toggle Try_static_cast</code>	166
8.7	Limitations in Pointer-to-Member Comparison	166
8.8	Using Templates.....	167
8.8.1	The Single-Copy Problem	167
8.8.2	Using COMDAT Sections to Merge Global Instantiations.....	168
8.8.3	Problems with Matching <code>const</code> Parameters	169

9

Data Representations	171
9.1 Data Types.....	171
9.1.1 Characters	171
9.1.2 Integers	172
9.1.3 Floating-Point Numbers	173
9.1.4 Pointers	174
9.1.5 Structures, Unions, and Bit Fields	174
9.2 Registers	174
9.3 Unsigned Expressions in Conditional Preprocessing Directives	174

10

Storage Mapping	175
10.1 Size and Alignment of Data Types.....	175
10.1.1 Size of enum Types.....	177
10.1.2 Size and Alignment of <code>struct</code> and <code>union</code> Types	177
10.2 Arrays	178
10.3 Bit Fields	178
10.3.1 How the Compiler Maps Bit Fields	179
10.4 Reversed Endianness	182
10.5 Aligning Data	182
10.5.1 Aligning Structure Members: <code>Pragma Align_members</code>	182

10.5.2	Aligning Data Structures: Pragma Pack	184
10.5.3	Aligning Structures and Saving Maximum Alignment: Pragma Push_align_members	184
10.5.4	Restoring Alignment: Pragma Pop_align_members	186
10.6	How the Compiler Maps Variables to Storage	186
10.7	Storage Mapping for C++ Class Objects	188
10.7.1	Virtual-Function Tables	188
10.7.2	Direct and Indirect Bases	189

11

PowerPC Run-Time Organization	195
11.1 Stack-Frame Layout	195
11.2 Register Usage and Naming Conventions	198
11.2.1 Registers Used in the Standard Calling Sequence	200
11.2.2 Functions Called During Signal Handling	201
11.2.3 Register Save Areas	201
11.3 Parameter Passing	202
11.4 Variable Argument Lists	202
11.5 Application Binary Interface	203
11.5.1 Overview of Global Offset Table and Procedure Linkage Table	203
11.5.2 Global Offset Table	203
11.5.3 Procedure Linkage Table	204
11.5.4 Small-Data Sections	205
11.5.5 Accessing Symbols with Assembler Identifier Attributes	210
11.5.6 Tag Support	211
11.6 Return Values	212
11.6.1 Returning Structures and Unions up to Eight Bytes in Size 213	
11.6.2 Values Returned in a Storage Buffer	213
11.7 Prolog and Epilog Code	213

12

Assembly-Language Communication	217
12.1 Overview	217
12.2 Coding Assembly Routines	218
12.3 Function-Naming Conventions in C	218
12.4 Function-Naming Conventions in C++	219
12.5 Calling Assembly Routines from C and C++	219

12.6	Calling C Functions from Assembly Routines	220
12.7	Sharing Variables Across Modules	221
13		
	Debugging and Diagnostic Tools.....	225
13.1	Debugger Support.....	225
13.2	Tracing Function Calls	226
13.3	Minimizing Symbolic Debug Information	227
13.3.1	How the Compiler Minimizes SDI.....	228
A		
	Generating List Files	233
A.1	Generating a Source Listing	233
A.2	Customizing the Listing with Pragmas Page, Skip, and Title... 233	
A.3	Listing Format	234
B		
	Configuring the Driver	237
B.1	Overview	237
B.2	User-Modifiable Configuration Variables.....	237
B.3	The Driver Configuration Language	240
B.3.1	Driver Command Reference	240
C		
	Using the Optional Inliner	245
C.1	Overview	245
C.2	Invoking the Inliner	246
C.2.1	How the Compiler Selects Routines for Inlining.....	247
C.2.2	Debugging Inlined Routines	247
C.3	Inliner Option Reference	248
C.4	How the Inliner Constructs Unique Names.....	250
C.5	Pragmas for Inlining	251
D		
	Developing Embedded Applications.....	253
D.1	PowerPC Models and Features Not Supported by High C/C++	253
D.2	Getting Started: Hardware and Software Considerations.....	254
D.2.1	Function Libraries.....	254
D.3	Start-Up Code for Executables: An Example.....	255

D.3.1	Link Order	257
D.3.2	Setting Module Initialization Priority: Pragma Initialization_level	258
D.3.3	Required Start-Up Operations	260
D.3.4	Special Uses for Start-Up Code	261
D.4	Storing crt1.o in an Archive Library	262
D.5	Writing Start-Up Code for Shared Libraries	262
D.6	Mapping Exception-Handling Code in Memory	264

E

Manual Template Instantiation.....		265
E.1	Overview.....	265
E.2	Preventing Duplicate Function Definitions with Toggle Auto_ func_instantiation	266
E.3	Class Templates and Duplicate Definitions.....	268
E.4	Selective Template Instantiation.....	270
E.5	Checking Template Usage: Toggle Print_template_usage... 272	

F

The Heap Manager	275
F.1 Overview	275
F.2 The Heap Chain	276
F.3 The Free Chain	278
F.4 Heap Integrity Checking	280

G

Loop Unrolling	281
G.1 Overview.....	281
G.2 Selecting Loops for Unrolling: Pragma Loop_factor.....	281
G.3 How the Compiler Unrolls Loops.....	282

H

Controlling Diagnostic Messages.....	285
H.1 Types of Diagnostic Messages.....	285
H.2 File I/O Errors	285
H.3 System Errors	286
H.4 User Error and Warning Messages	287
H.4.1 Lexical Error Messages	287
H.4.2 Syntactic Error Messages	288

:

H.4.3	Constraint Error and Warning Messages.....	288
H.5	Changing the Message Format	289
H.5.1	Predefined Message Formats.....	290
H.6	Controlling Warning Messages	290
H.6.1	Controlling Groups of Warnings by Level	291
H.6.2	Controlling Individual Warnings by Number.....	291
Index	295