Sourcery CodeBench Lite

MIPS GNU/Linux

Sourcery CodeBench Lite 2013.05-66

Getting Started

Sourcery CodeBench Lite: MIPS GNU/Linux: Sourcery CodeBench Lite 2013.05-66: Getting Started

Mentor Graphics, Inc.

Abstract

This guide explains how to install and build applications with Sourcery CodeBench Lite, CodeSourcery's customized and validated version of the GNU Toolchain. Sourcery CodeBench Lite includes everything you need for application development, including C and C++ compilers, assemblers, linkers, and libraries.

When you have finished reading this guide, you will know how to use Sourcery CodeBench from the command line.

Table of Contents

Preface

This preface introduces the Sourcery CodeBench Lite Getting Started guide. It explains the structure of this guide and describes the documentation conventions used.

1. Intended Audience

This guide is written for people who will install and/or use Sourcery CodeBench Lite. This guide provides a step-by-step guide to installing Sourcery CodeBench Lite and to building simple applications. Parts of this document assume that you have some familiarity with using the command-line interface.

2. Organization

This document is organized into the following chapters and appendices:

Chapter 1, "Quick Start"	This chapter includes a brief checklist to follow when installing and using Sourcery CodeBench Lite for the first time. You may use this chapter as an abbreviated guide to the rest of this manual.
Chapter 2, "Installation and Configuration"	This chapter describes how to download, install and configure Sourcery CodeBench Lite. This section describes the available installation options and explains how to set up your environment so that you can build applications.
Chapter 3, "Sourcery CodeBench Lite for MIPS GNU/Linux"	This chapter contains information about using Sourcery CodeBench Lite that is specific to MIPS GNU/Linux targets. You should read this chapter to learn how to best use Sourcery CodeBench Lite on your target system.
Chapter 4, "Using Sourcery CodeBench from the Command Line"	This chapter explains how to build applications with Sourcery CodeBench Lite using the command line. In the process of reading this chapter, you will build a simple application that you can use as a model for your own programs.
Chapter 5, "Sourcery CodeBench Debug Sprite"	This chapter describes the use of the Sourcery CodeBench Debug Sprite for remote debugging. The Sprite is provided for debugging of the Linux kernel on the target board. This chapter includes information about the debugging devices and boards supported by the Sprite for MIPS GNU/Linux.
Chapter 6, "Next Steps with Sourcery CodeBench"	This chapter describes where you can find additional documentation and information about using Sourcery CodeBench Lite and its components. It also provides information about Sourcery CodeBench subscriptions. CodeSourcery customers with Sourcery CodeBench subscriptions receive comprehensive support for Sourcery CodeBench.
Appendix A, "Sourcery CodeBench Lite Release Notes"	This appendix contains information about changes in this release of Sourcery CodeBench Lite for MIPS GNU/Linux. You should read through these notes to learn about new features and bug fixes.
Appendix B, "Sourcery CodeBench Lite Licenses"	This appendix provides information about the software licenses that apply to Sourcery CodeBench Lite. Read this appendix to understand your legal rights and obligations as a user of Sourcery CodeBench Lite.

3. Typographical Conventions

The following typographical conventions are used in this guide:

`> command arg ...`	A command, typed by the user, and its output. The “>” character is the command prompt.
`command`	The name of a program, when used in a sentence, rather than in literal input or output.
`literal`	Text provided to or received from a computer program.
`placeholder`	Text that should be replaced with an appropriate value when typing a command.
`\`	At the end of a line in command or program examples, indicates that a long line of literal input or output continues onto the next line in the document.

Chapter 1
Quick Start

This chapter includes a brief checklist to follow when installing and using Sourcery CodeBench Lite for the first time. You may use this chapter as an abbreviated guide to the rest of this manual.

Sourcery CodeBench Lite for MIPS GNU/Linux is intended for developers working on embedded GNU/Linux applications. It may also be used for Linux kernel development and debugging, or to build a GNU/Linux distribution.

Follow the steps given in this chapter to install Sourcery CodeBench Lite and build and run your first application program. The checklist given here is not a tutorial and does not include detailed instructions for each step; however, it will help guide you to find the instructions and reference information you need to accomplish each step. Note that this checklist is also oriented towards application debugging rather than kernel debugging.

You can find additional details about the components, libraries, and other features included in this version of Sourcery CodeBench Lite in Chapter 3, "Sourcery CodeBench Lite for MIPS GNU/Linux".

1.1. Installation and Set-Up

Install Sourcery CodeBench Lite on your host computer. You may download an installer package from the Sourcery CodeBench web site[1], or you may have received an installer on CD. The installer is an executable program that pops up a window on your computer and leads you through a series of dialogs to configure your installation. When the installation is complete, it offers to launch the Getting Started guide. For more information about installing Sourcery CodeBench Lite, including host system requirements and tips to set up your environment after installation, refer to Chapter 2, "Installation and Configuration".

1.2. Configuring Sourcery CodeBench Lite for the Target System

Identify your target libraries. Sourcery CodeBench Lite supports libraries optimized for different targets. Libraries are selected automatically by the linker, depending on the processor and other options you have specified. Refer to Section 3.2, "Library Configurations" for details.

Install runtime libraries on your target machine. In order to run programs built with the Sourcery CodeBench runtime libraries on target hardware, you must install these libraries, the Sourcery CodeBench dynamic linker, and other runtime support files -- collectively referred to as the *sysroot* -- on your GNU/Linux target system. Typically, this involves either using third-party tools to build a complete root filesystem including the Sourcery CodeBench sysroot, or using special commands when linking or running your program so it can find the sysroot installed in another location on the target. Refer to Section 3.5, "Using Sourcery CodeBench Lite on GNU/Linux Targets" for full discussion of these options. You can skip this step if you plan to run your program with the QEMU user-space emulator on a Linux host.

1.3. Building Your Program

Build your program with Sourcery CodeBench command-line tools. Create a simple test program, and follow the directions in Chapter 4, "Using Sourcery CodeBench from the Command Line" to compile and link it using Sourcery CodeBench Lite.

1.4. Running and Debugging Your Program

The steps to run or debug your program depend on your target system and how it is configured. Choose the appropriate method for your target.

[1] http://go.mentor.com/codebench/

Debug your program in the QEMU user-space emulator. If you have installed Sourcery CodeBench on a GNU/Linux host, you can use the QEMU user-space emulator to debug your program. This is an easy way to try out Sourcery CodeBench Lite without target hardware. Refer to Section 4.3, "Running Applications with QEMU" for instructions on using QEMU.

Run your program on the MIPS GNU/Linux target. To run a program using the included Sourcery CodeBench libraries, you must install the sysroot on the target, as previously discussed. Copy the executable for your program to the target system. The method you use for launching your program depends on how you have installed the libraries and built your program. In some cases, you may need to invoke the Sourcery CodeBench dynamic linker explicitly. Refer to Section 3.5, "Using Sourcery CodeBench Lite on GNU/Linux Targets" for details.

Debug your program on the target using GDB server. You can use GDB server on a remote target to debug your program. When debugging a program that uses the included Sourcery CodeBench libraries, you must use the `gdbserver` executable included in the sysroot, and similar issues with respect to the dynamic linker as discussed previously apply. See Section 3.6, "Using GDB Server for Debugging" for detailed instructions. Once you have started GDB server on the target, you can connect to it from the debugger on your host system. Refer to Section 4.4, "Running Applications from GDB" for instructions on remote debugging from command-line GDB.

Chapter 2
Installation and Configuration

This chapter explains how to install Sourcery CodeBench Lite. You will learn how to:

1. Verify that you can install Sourcery CodeBench Lite on your system.
2. Download the appropriate Sourcery CodeBench Lite installer.
3. Install Sourcery CodeBench Lite.
4. Configure your environment so that you can use Sourcery CodeBench Lite.

2.1. Terminology

Throughout this document, the term *host system* refers to the system on which you run Sourcery CodeBench while the term *target system* refers to the system on which the code produced by Sourcery CodeBench runs. The target system for this version of Sourcery CodeBench is `mips-linux-gnu`.

If you are developing a workstation or server application to run on the same system that you are using to run Sourcery CodeBench, then the host and target systems are the same. On the other hand, if you are developing an application for an embedded system, then the host and target systems are probably different.

2.2. System Requirements

2.2.1. Host Operating System Requirements

This version of Sourcery CodeBench supports the following host operating systems and architectures:

- Microsoft Windows XP (SP1), Windows Vista, and Windows 7 systems using IA32, AMD64, and Intel 64 processors.
- GNU/Linux systems using IA32, AMD64, or Intel 64 processors, including Debian 5 (and later), Red Hat Enterprise Linux 5 (and later), SuSE Enterprise Linux 10 (and later), and Ubuntu 8.04 (and later).

Sourcery CodeBench is built as a 32-bit application. Therefore, even when running on a 64-bit host system, Sourcery CodeBench requires 32-bit host libraries. If these libraries are not already installed on your system, you must install them before installing and using Sourcery CodeBench Lite. Consult your operating system documentation for more information about obtaining these libraries.

Installing on Ubuntu and Debian GNU/Linux Hosts

The Sourcery CodeBench graphical installer is incompatible with the `dash` shell, which is the default `/bin/sh` for recent releases of the Ubuntu and Debian GNU/Linux distributions. To install Sourcery CodeBench Lite on these systems, you must make `/bin/sh` a symbolic link to one of the supported shells: `bash`, `csh`, `tcsh`, `zsh`, or `ksh`.

For example, on Ubuntu systems, the recommended way to do this is:

```
> sudo dpkg-reconfigure -plow dash
Install as /bin/sh? No
```

This is a limitation of the installer and uninstaller only, not of the installed Sourcery CodeBench Lite toolchain.

2.2.2. Host Hardware Requirements

The amount of disk space required for a complete Sourcery CodeBench Lite installation directory depends on the host operating system and the number of target libraries included. When you start the graphical installer, it checks whether there is sufficient disk space before beginning to install. Note that the graphical installer also requires additional temporary disk space during the installation process. On Microsoft Windows hosts, the installer uses the location specified by the `TEMP` environment variable for these temporary files. If there is not enough free space on that volume, the installer prompts for an alternate location. On Linux hosts, the installer puts temporary files in the directory specified by the `IATEMPDIR` environment variable, or `/tmp` if that is not set.

2.2.3. Target System Requirements

See Chapter 3, “Sourcery CodeBench Lite for MIPS GNU/Linux” for requirements that apply to the target system.

2.3. Downloading an Installer

If you have received Sourcery CodeBench Lite on a CD, or other physical media, then you do not need to download an installer. You may skip ahead to Section 2.4, “Installing Sourcery CodeBench Lite”.

You can download Sourcery CodeBench Lite from the Sourcery CodeBench web site[1]. This free version of Sourcery CodeBench, which is made available to the general public, does not include all the functionality of CodeSourcery's product releases. If you prefer, you may instead purchase or register for an evaluation of CodeSourcery's product toolchains at the Sourcery CodeBench Portal[2].

Once you have navigated to the appropriate web site, download the installer that corresponds to your host operating system. For Microsoft Windows systems, the Sourcery CodeBench installer is provided as an executable with the `.exe` extension. For GNU/Linux systems Sourcery CodeBench Lite is provided as an executable installer package with the `.bin` extension. You may also install from a compressed archive with the `.tar.bz2` extension.

On Microsoft Windows systems, save the installer to the desktop. On GNU/Linux systems, save the download package in your home directory.

2.4. Installing Sourcery CodeBench Lite

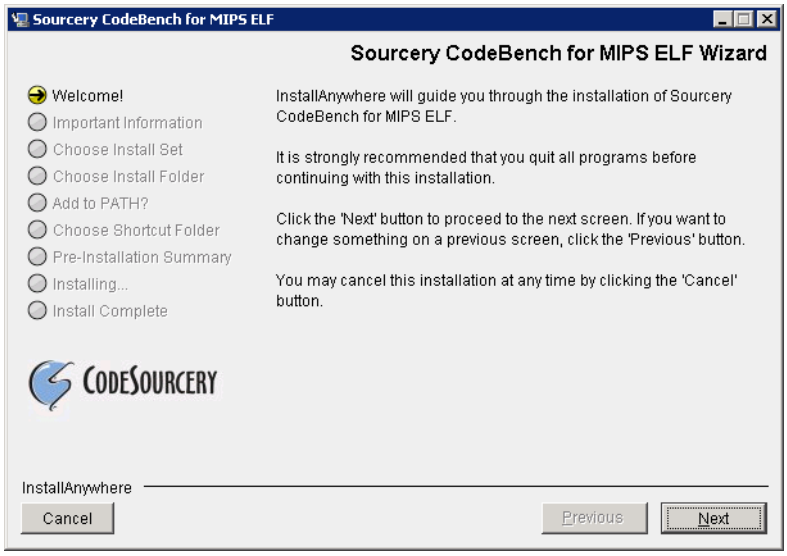
The method used to install Sourcery CodeBench Lite depends on your host system and the kind of installation package you have downloaded.

2.4.1. Using the Sourcery CodeBench Lite Installer on Microsoft Windows

If you have received Sourcery CodeBench Lite on CD, insert the CD in your computer. On most computers, the installer then starts automatically. If your computer has been configured not to automatically run CDs, open `My Computer`, and double click on the CD. If you downloaded Sourcery CodeBench Lite, double-click on the installer.

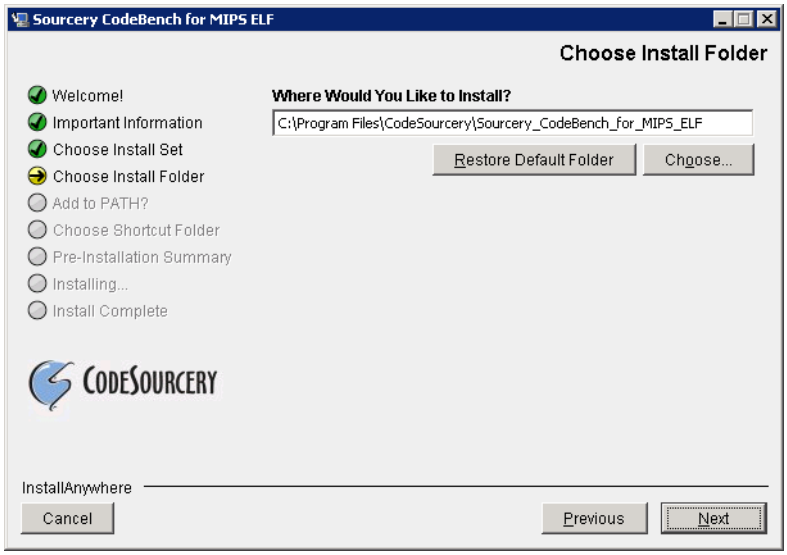
After the installer starts, follow the on-screen dialogs to install Sourcery CodeBench Lite. The installer is intended to be self-explanatory and on most pages the defaults are appropriate.

[1] http://go.mentor.com/codebench/
[2] https://sourcery.mentor.com/GNUToolchain/

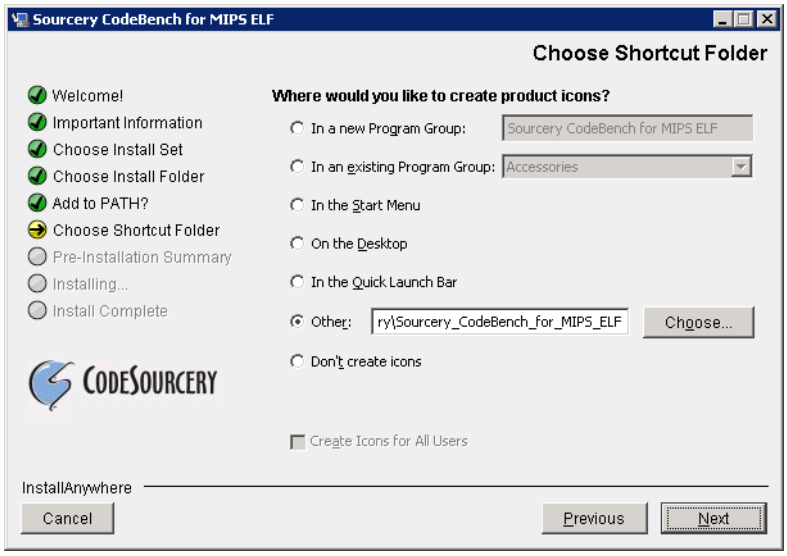


Running the Installer. The graphical installer guides you through the steps to install Sourcery CodeBench Lite.

You may want to change the install directory pathname and customize the shortcut installation.

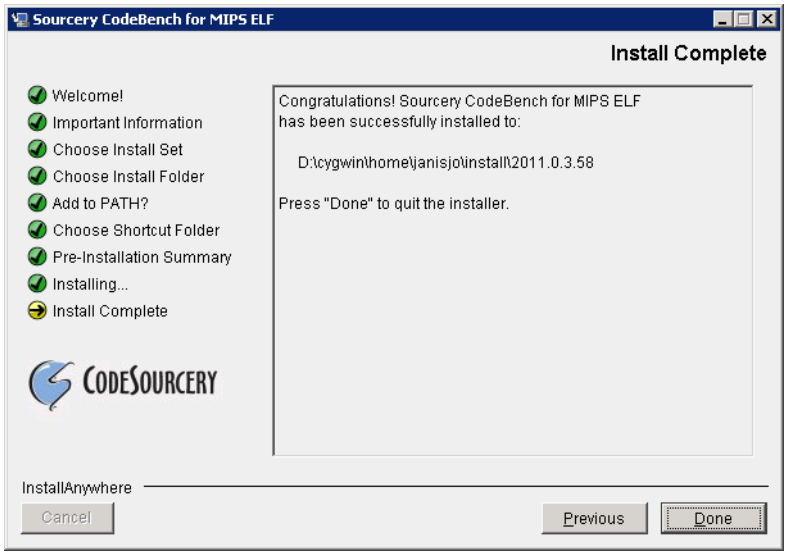


Choose Install Folder. Select the pathname to your install directory.



Choose Shortcut Folder. You can customize where the installer creates shortcuts for quick access to Sourcery CodeBench Lite.

When the installer has finished, it asks if you want to launch a viewer for the Getting Started guide. Finally, the installer displays a summary screen to confirm a successful install before it exits.



Install Complete. You should see a screen similar to this after a successful install.

If you prefer, you can run the installer in console mode rather than using the graphical interface. To do this, invoke the installer with the `-i console` command-line option. For example:

```
> /path/to/package.exe -i console
```

2.4.2. Using the Sourcery CodeBench Lite Installer on GNU/Linux Hosts

Start the graphical installer by invoking the executable shell script:

```
> /bin/sh ./path/to/package.bin
```

After the installer starts, follow the on-screen dialogs to install Sourcery CodeBench Lite. For additional details on running the installer, see the discussion and screen shots in the Microsoft Windows section above.

If you prefer, or if your host system does not run the X Window System, you can run the installer in console mode rather than using the graphical interface. To do this, invoke the installer with the `-i console` command-line option. For example:

```
> /bin/sh ./path/to/package.bin -i console
```

2.4.3. Installing Sourcery CodeBench Lite from a Compressed Archive

You do not need to be a system administrator to install Sourcery CodeBench Lite from a compressed archive. You may install Sourcery CodeBench Lite using any user account and in any directory to which you have write access. This guide assumes that you have decided to install Sourcery CodeBench Lite in the `$HOME/CodeSourcery` subdirectory of your home directory and that the filename of the package you have downloaded is `/path/to/package.tar.bz2`. After installation the toolchain will be in `$HOME/CodeSourcery/sourceryg++-2013.05`.

First, uncompress the package file:

```
> bunzip2 /path/to/package.tar.bz2
```

Next, create the directory in which you wish to install the package:

```
> mkdir -p $HOME/CodeSourcery
```

Change to the installation directory:

```
> cd $HOME/CodeSourcery
```

Unpack the package:

```
> tar xf /path/to/package.tar
```

2.5. Installing Sourcery CodeBench Lite Updates

If you have already installed an earlier version of Sourcery CodeBench Lite for MIPS GNU/Linux on your system, it is not necessary to uninstall it before using the installer to unpack a new version in the same location. The installer detects that it is performing an update in that case.

If you are installing an update from a compressed archive, it is recommended that you remove any previous installation in the same location, or install in a different directory.

Note that the names of the Sourcery CodeBench commands for the MIPS GNU/Linux target all begin with `mips-linux-gnu`. This means that you can install Sourcery CodeBench for multiple target systems in the same directory without conflicts.

2.6. Setting up the Environment

As with the installation process itself, the steps required to set up your environment depend on your host operating system.

2.6.1. Setting up the Environment on Microsoft Windows Hosts

2.6.1.1. Setting the PATH

The graphical installer for Sourcery CodeBench Lite does this setup for you, however it may not take effect until you next log in.

In order to use the Sourcery CodeBench tools from the command line, you should add them to your PATH. In the instructions that follow, replace *installdir* with the full pathname of your Sourcery CodeBench Lite installation directory, including the drive letter.

To set the PATH on a Microsoft Windows Vista system, use the following command in a cmd.exe shell:

```
> setx PATH "%PATH%;installdir\bin"
```

To set the PATH on a system running Microsoft Windows 7, from the desktop bring up the Start menu and right click on Computer. Select Properties and click on Advanced system settings. Go to the Advanced tab, then click on the Environment Variables button. Select the PATH variable and click Edit. Add the string ;*installdir*\bin to the end, and click OK.

To set the PATH on older versions of Microsoft Windows, from the desktop bring up the Start menu and right click on My Computer. Select Properties, go to the Advanced tab, then click on the Environment Variables button. Select the PATH variable and click the Edit. Add the string ;*installdir*\bin to the end, and click OK.

You can verify that your PATH is set up correctly by starting a new cmd.exe shell and running:

```
> mips-linux-gnu-gcc -v
```

Verify that the last line of the output contains: Sourcery CodeBench Lite 2013.05-66.

2.6.1.2. Working with Cygwin

Sourcery CodeBench Lite does not require Cygwin or any other UNIX emulation environment. You can use Sourcery CodeBench directly from the Windows command shell. You can also use Sourcery CodeBench from within the Cygwin environment, if you prefer.

The Cygwin emulation environment translates Windows path names into UNIX path names. For example, the Cygwin path /home/user/hello.c corresponds to the Windows path c:\cygwin\home\user\hello.c. Because Sourcery CodeBench is not a Cygwin application, it does not, by default, recognize Cygwin paths.

If you are using Sourcery CodeBench from Cygwin, you should set the CYGPATH environment variable. If this environment variable is set, Sourcery CodeBench Lite automatically translates Cygwin path names into Windows path names. To set this environment variable, type the following command in a Cygwin shell:

```
> export CYGPATH=cygpath
```

To resolve Cygwin path names, Sourcery CodeBench relies on the cygpath utility provided with Cygwin. You must provide Sourcery CodeBench with the full path to cygpath if cygpath is not in your PATH. For example:

```
> export CYGPATH=c:/cygwin/bin/cygpath
```

directs Sourcery CodeBench Lite to use `c:/cygwin/bin/cygpath` as the path conversion utility. The value of `CYGPATH` must be an ordinary Windows path, not a Cygwin path.

2.6.2. Setting up the Environment on GNU/Linux Hosts

The graphical installer for Sourcery CodeBench Lite does this setup for you, however it may not take effect until you next log in.

Before using Sourcery CodeBench Lite you should add it to your `PATH`. The command you must use varies with the particular command shell that you are using. If you are using the C Shell (`csh` or `tcsh`), use the command:

```
> setenv PATH installdir/bin:$PATH
```

If you are using Bourne Shell (`sh`), the Korn Shell (`ksh`), or another shell, use:

```
> PATH=installdir/bin:$PATH
> export PATH
```

If you are not sure which shell you are using, try both commands. In both cases, replace *`installdir`* with the full pathname of your Sourcery CodeBench Lite installation directory.

You may also wish to set the `MANPATH` environment variable so that you can access the Sourcery CodeBench manual pages, which provide additional information about using Sourcery CodeBench. To set the `MANPATH` environment variable, follow the same steps shown above, replacing `PATH` with `MANPATH`, and `bin` with `share/doc/mips-mips-linux-gnu/man`.

You can test that your `PATH` is set up correctly by running the following command:

```
> mips-linux-gnu-gcc -v
```

Verify that the last line of the output contains: `Sourcery CodeBench Lite 2013.05-66`.

2.7. Uninstalling Sourcery CodeBench Lite

The method used to uninstall Sourcery CodeBench Lite depends on the method you originally used to install it. If you have modified any files in the installation it is recommended that you back up these changes. The uninstall procedure may remove the files you have altered. In particular, the `mips-linux-gnu` directory located in the install directory will be removed entirely by the uninstaller.

2.7.1. Using the Sourcery CodeBench Lite Uninstaller on Microsoft Windows

You should use the provided uninstaller to remove a Sourcery CodeBench Lite installation originally created by the graphical installer. Start the graphical uninstaller by invoking the Uninstall executable located in your installation directory, or use the uninstall shortcut created during installation. After the uninstaller starts, follow the on-screen dialogs to uninstall Sourcery CodeBench Lite.

You can run the uninstaller in console mode, rather than using the graphical interface, by invoking the Uninstall executable found in your Sourcery CodeBench Lite installation directory with the `-i console` command-line option.

To uninstall third-party drivers bundled with Sourcery CodeBench Lite, first disconnect the associated hardware device. Then use `Uninstall a program` (Vista and newer) or `Add or Remove`

`Programs` (older versions of Windows) to remove the drivers separately. Depending on the device, you may need to reboot your computer to complete the driver uninstall.

2.7.2. Using the Sourcery CodeBench Lite Uninstaller on GNU/Linux

You should use the provided uninstaller to remove a Sourcery CodeBench Lite installation originally created by the executable installer script. Start the graphical uninstaller by invoking the executable Uninstall shell script located in your installation directory. After the uninstaller starts, follow the on-screen dialogs to uninstall Sourcery CodeBench Lite.

You can run the uninstaller in console mode, rather than using the graphical interface, by invoking the Uninstall script with the `-i console` command-line option.

2.7.3. Uninstalling a Compressed Archive Installation

If you installed Sourcery CodeBench Lite from a `.tar.bz2` file, you can uninstall it by manually deleting the installation directory created in the install procedure.

Chapter 3
Sourcery CodeBench Lite for MIPS GNU/Linux

This chapter contains information about features of Sourcery CodeBench Lite that are specific to MIPS GNU/Linux targets. You should read this chapter to learn how to best use Sourcery CodeBench Lite on your target system.

3.1. Included Components and Features

This section briefly lists the important components and features included in Sourcery CodeBench Lite for MIPS GNU/Linux, and tells you where you may find further information about these features.

Component	Version	Notes
GNU programming tools		
GNU Compiler Collection	4.7.3	Separate manual included.
GNU Binary Utilities	2.23.52	Includes assembler, linker, and other utilities. Separate manuals included.
Debugging support and simulators		
GNU Debugger	7.4.50	Separate manual included.
Sourcery CodeBench Debug Sprite	2013.05-66	Provided for kernel debugging only. See Chapter 5, “Sourcery CodeBench Debug Sprite”.
GDB Server	N/A	Included with GDB. See Section 3.6, “Using GDB Server for Debugging”.
QEMU Emulator	1.0.50	User-space emulation is supported on Linux hosts only. See Section 4.3, “Running Applications with QEMU”.
Target libraries		
GNU C Library	2.17	Separate manual included.
uClibc C Library	0.9.30	
Linux Kernel Headers	3.8.2	
OpenMP	N/A	
Other utilities		
GNU Make	N/A	Build support on Windows hosts.
GNU Core Utilities	N/A	Build support on Windows hosts.
GNU/Linux Prelinker	N/A	See Section 4.5, “Using the Prelinker”.

3.2. Library Configurations

Sourcery CodeBench Lite for MIPS GNU/Linux includes the following library configuration.

MIPS32 revision 2 - Big-Endian, O32	
Command-line option(s):	default
Sysroot subdirectory:	`./`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld.so.1`

MIPS64 revision 2 - Big-Endian, N64	
Command-line option(s):	`-mabi=64`
Sysroot subdirectory:	`./`
Library subdirectories:	`lib64/` and `usr/lib64/`
Dynamic linker:	`lib64/ld.so.1`

MIPS32 revision 2 - Little-Endian, O32	
Command-line option(s):	`-EL`
Sysroot subdirectory:	`el/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld.so.1`

MIPS64 revision 2 - Little-Endian, N64	
Command-line option(s):	`-EL -mabi=64`
Sysroot subdirectory:	`el/`
Library subdirectories:	`lib64/` and `usr/lib64/`
Dynamic linker:	`lib64/ld.so.1`

MIPS32 revision 2 - Big-Endian, Soft-Float, O32	
Command-line option(s):	`-msoft-float`
Sysroot subdirectory:	`soft-float/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld.so.1`

MIPS64 revision 2 - Big-Endian, Soft-Float, N64	
Command-line option(s):	`-msoft-float -mabi=64`
Sysroot subdirectory:	`soft-float/`
Library subdirectories:	`lib64/` and `usr/lib64/`
Dynamic linker:	`lib64/ld.so.1`

MIPS32 revision 2 - Little-Endian, Soft-Float, O32	
Command-line option(s):	`-EL -msoft-float`
Sysroot subdirectory:	`soft-float/el/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld.so.1`

MIPS64 revision 2 - Little-Endian, Soft-Float, N64	
Command-line option(s):	`-EL -msoft-float -mabi=64`
Sysroot subdirectory:	`soft-float/el/`
Library subdirectories:	`lib64/` and `usr/lib64/`
Dynamic linker:	`lib64/ld.so.1`

MIPS32 revision 2 - Big-Endian, 2008 NaN, O32	
Command-line option(s):	`-mnan2008`
Sysroot subdirectory:	`nan2008/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld-linux-mipsn8.so.1`

MIPS32 revision 2 - Little-Endian, 2008 NaN, O32	
Command-line option(s):	`-EL -mnan2008`
Sysroot subdirectory:	`nan2008/el/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld-linux-mipsn8.so.1`

MIPS32 revision 2 - Big-Endian, O32, microMIPS	
Command-line option(s):	`-mmicromips`
Sysroot subdirectory:	`micromips/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld.so.1`

MIPS32 revision 2 - Little-Endian, O32, microMIPS	
Command-line option(s):	`-mmicromips -EL`
Sysroot subdirectory:	`micromips/el/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld.so.1`

MIPS32 revision 2 - Big-Endian, Soft-Float, O32, microMIPS	
Command-line option(s):	`-mmicromips -msoft-float`
Sysroot subdirectory:	`micromips/soft-float/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld.so.1`

MIPS32 revision 2 - Little-Endian, Soft-Float, O32, microMIPS	
Command-line option(s):	`-mmicromips -EL -msoft-float`
Sysroot subdirectory:	`micromips/soft-float/el/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld.so.1`

MIPS32 revision 2 - Big-Endian, O32, MIPS16	
Command-line option(s):	`-mips16`
Sysroot subdirectory:	`mips16/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld.so.1`

MIPS32 revision 2 - Little-Endian, O32, MIPS16	
Command-line option(s):	`-mips16 -EL`
Sysroot subdirectory:	`mips16/el/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld.so.1`

MIPS32 revision 2 - Big-Endian, Soft-Float, O32, MIPS16	
Command-line option(s):	`-mips16 -msoft-float`
Sysroot subdirectory:	`mips16/soft-float/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld.so.1`

MIPS32 revision 2 - Little-Endian, Soft-Float, O32, MIPS16	
Command-line option(s):	`-mips16 -EL -msoft-float`
Sysroot subdirectory:	`mips16/soft-float/el/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld.so.1`

MIPS32 revision 2 - Big-Endian, 2008 NaN, O32, MIPS16	
Command-line option(s):	`-mips16 -mnan2008`
Sysroot subdirectory:	`mips16/nan2008/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld-linux-mipsn8.so.1`

MIPS32 revision 2 - Little-Endian, 2008 NaN, O32, MIPS16	
Command-line option(s):	`-mips16 -EL -mnan2008`
Sysroot subdirectory:	`mips16/nan2008/el/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld-linux-mipsn8.so.1`

MIPS32 revision 2 - uClibc, Big-Endian, O32	
Command-line option(s):	`-muclibc`
Sysroot subdirectory:	`uclibc/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld-uClibc.so.0`

MIPS32 revision 2 - uClibc, Little-Endian, O32	
Command-line option(s):	`-muclibc -EL`
Sysroot subdirectory:	`uclibc/el/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld-uClibc.so.0`

MIPS32 revision 2 - uClibc, Big-Endian, Soft-Float, O32	
Command-line option(s):	`-muclibc -msoft-float`
Sysroot subdirectory:	`uclibc/soft-float/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld-uClibc.so.0`

MIPS32 revision 2 - uClibc, Little-Endian, Soft-Float, O32	
Command-line option(s):	`-muclibc -EL -msoft-float`
Sysroot subdirectory:	`uclibc/soft-float/el/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld-uClibc.so.0`

MIPS32 revision 2 - uClibc, Big-Endian, 2008 NaN, O32	
Command-line option(s):	`-muclibc -mnan2008`
Sysroot subdirectory:	`uclibc/nan2008/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld-uClibc-mipsn8.so.0`

MIPS32 revision 2 - uClibc, Little-Endian, 2008 NaN, O32	
Command-line option(s):	`-muclibc -EL -mnan2008`
Sysroot subdirectory:	`uclibc/nan2008/el/`
Library subdirectories:	`lib/` and `usr/lib/`
Dynamic linker:	`lib/ld-uClibc-mipsn8.so.0`

Sourcery CodeBench includes copies of run-time libraries that have been built with optimizations for different target architecture variants or other sets of build options. Each such set of libraries is referred to as a *multilib*. When you link a target application, Sourcery CodeBench selects the multilib matching the build options you have selected.

Each multilib corresponds to a *sysroot* directory which contains the files that should be installed on the target system. The sysroot contains the dynamic linker used to run your applications on the target as well as the libraries. Refer to Section 3.5, "Using Sourcery CodeBench Lite on GNU/Linux Targets" for instructions on how to install and use these support files on your target GNU/Linux system. You can find the sysroot directories provided with Sourcery CodeBench in the `mips-linux-gnu/libc` directory of your installation. In the tables below, the dynamic linker pathname is given relative to the corresponding sysroot.

3.3. Target Architectures

By default, Sourcery CodeBench Lite for MIPS GNU/Linux generates code for MIPS32r2 processors. If you wish to generate code for another MIPS processor, you must use an appropriate `-march` option when you build your application. Refer to the GCC manual for additional information about supported targets.

3.4. Target Kernel Requirements

The GNU C library supplied with this version of Sourcery CodeBench Lite requires that Linux kernel version 2.6.16 or later be installed on the target in order to run applications.

To support hardware watchpoints with `gdbserver`, Linux 2.6.28 or later is required.

3.5. Using Sourcery CodeBench Lite on GNU/Linux Targets

In order to run and debug programs produced by Sourcery CodeBench on a GNU/Linux target, you must install runtime support files on the target. You may also need to set appropriate build options so that your executables can find the correct dynamic linker and libraries at runtime.

The runtime support files, referred to as the *sysroot*, are found in the `mips-linux-gnu/libc` directory of your Sourcery CodeBench Lite installation. The sysroot consists of the contents of the `etc`, `lib`, `sbin`, and `usr` directories. There may be other directories in `mips-linux-gnu/libc` that contain additional sysroots customized for particular combinations of command-line compiler flags, or *multilibs*. Refer to Section 3.2, "Library Configurations" for a list of the included multilibs in this version of Sourcery CodeBench Lite, and the corresponding sysroot directory pathnames.

Note for Windows Host Users

The sysroots provided in Windows host packages for Sourcery CodeBench are not directly usable on the GNU/Linux target because of differences between the Windows and GNU/Linux file systems. Some files that are hard links, or copies, in the sysroot as installed on the Windows file system should be symbolic links on the GNU/Linux target. Additionally, some files in the sysroot that should be marked executable on the GNU/Linux target are not marked executable on Windows. If you intend to use the sysroot provided with Sourcery CodeBench on a Windows host system as the basis for your GNU/Linux target filesystem, you must correct these issues after copying the sysroot to the target.

You have these choices for installing the sysroot on the target:

- You can install the files in the filesystem root on the target (that is, installing the files directly in `/etc/`, `/lib/`, and so on). All applications on the target then automatically use the Sourcery CodeBench libraries. This method is primarily useful when you are building a GNU/Linux root filesystem from scratch. If your target board already has a GNU/Linux filesystem installed, overwriting the existing C library files is not recommended, as this may break other applications on your system, or cause it to fail to boot.

- You can install the sysroot in an alternate location and build your application with the `-rpath` and `--dynamic-linker` linker options to specify the sysroot location.

- You can install the sysroot in an alternate location and explicitly invoke your application through the dynamic linker to specify the sysroot location. If you are just getting started with Sourcery CodeBench Lite, this may be the easiest way to get your application running, but this method does not support use of the debugger. In addition, this method only works with programs linked with the GNU C Library (glibc), not uClibc.

Setting the environment variable `LD_LIBRARY_PATH` on the target is not sufficient, since executables produced by Sourcery CodeBench depend on the Sourcery CodeBench dynamic linker included in the sysroot as well as the Sourcery CodeBench runtime libraries.

3.5.1. Installing the Sysroot

If you are modifying an existing system, rather than creating a new system from scratch, you should place the sysroot files in a new directory, rather than in the root directory of your target system.

If you choose to overwrite your existing C library, you may not be able to boot your system. You should back up your existing system before overwriting the C library and ensure that you can restore the backup even with your system offline.

The next step is to identify the correct sysroot subdirectory in the Sourcery CodeBench Lite install directory on your host system. The sysroot you copy to the target must be the one that corresponds to the linker options you are using to build your applications. The tables in Section 3.2, "Library Configurations" tell you which sysroot subdirectories correspond to which sets of command-line options. From the command line, you can identify the appropriate sysroot for your program by invoking the compiler with `-print-sysroot` added to your other build options. This causes GCC to print the host sysroot pathname and exit.

The mechanism you use for copying the sysroot to your target board depends on its hardware and software configuration. You may be able to use FTP or SSH with a server already running on your target. If your target board does not have networking configured, you may be able to copy files using an SD card or USB memory stick, or via a file transfer utility over a serial line. The instructions that come with your board may include specific suggestions.

When running Sourcery CodeBench on a GNU/Linux host, as an alternative to copying files to the target system, you may be able to NFS-mount the Sourcery CodeBench Lite installation directory from your host system on the target system. It is especially convenient for debugging if you can make the sysroot pathname on the target system be identical to that on the GNU/Linux host system; refer to Section 3.6.3, "Setting the Sysroot in the Debugger" for further discussion of this issue.

Otherwise, you must copy files from the appropriate sysroot subdirectory in the `mips-linux-gnu/libc` directory of your Sourcery CodeBench Lite install to the target system. In many cases, you do not need to copy all of the files in the sysroot. For example, the `usr/include` subdirectory contains files that are only needed if you will actually be running the compiler on your target system. You do not need these files for non-native compilers. You also do not need any `.o` or `.a` files; these are used by the compiler when linking programs, but are not needed to run programs. You should definitely copy all `.so` files and the executable files in `usr/bin` and `sbin`.

3.5.2. Using Linker Options to Specify the Sysroot Location

If you have installed the sysroot on the target in a location other than the file system root, you can use the `-rpath` and `--dynamic-linker` linker options to specify the sysroot location.

If you are using Sourcery CodeBench from the command line, follow these steps:

1. First find the correct sysroot, dynamic linker, and library subdirectory for your selected multilib. Refer to Section 3.2, "Library Configurations". In the following steps, *`sysroot`* is the absolute path to the directory on the target where you have installed the sysroot corresponding to your selected multilib. *`libsubdir`* specifies the library subdirectories relative to the sysroot. *`ldso`* is the filename of the dynamic linker inside the respective *`libsubdir`*.

2. When invoking `mips-linux-gnu-gcc` to link your executable, include the command-line options:

```
-Wl,-rpath=sysroot/libsubdir:sysroot/usr/libsubdir \
-Wl,--dynamic-linker=sysroot/libsubdir/ldso
```

   For the default multilib, this becomes:

```
-Wl,-rpath=sysroot/lib:sysroot/usr/lib \
-Wl,--dynamic-linker=sysroot/lib/ld.so.1
```

3. Copy the executable to the target and execute it normally.

Note that if you specify an incorrect path for `--dynamic-linker`, the common failure mode seen when running your application on the target is similar to

```
> ./factorial
./factorial: No such file or directory
```

or

```
> ./factorial
./factorial: bad ELF interpreter: No such file or directory
```

This can be quite confusing since it appears from the error message as if it is the `./factorial` executable that is missing rather than the dynamic linker it references.

3.5.3. Specifying the Sysroot Location at Runtime

You can invoke the Sourcery CodeBench dynamic linker on the target to run your application without having to compile it with specific linker options. Note that this method of specifying the sysroot is specific to the GNU C Library (glibc) and does not work if you have linked your application with uClibc instead.

To do this, follow these steps:

1. Build your application on the host, without any additional linker options, and copy the executable to your target system.

2. First find the correct sysroot, dynamic linker, and library subdirectory for your selected multilib. Refer to Section 3.2, “Library Configurations”. In the following steps, *`sysroot`* is the absolute path to the directory on the target where you have installed the sysroot corresponding to your selected multilib. *`libsubdir`* specifies the library subdirectories relative to the sysroot. *`ldso`* is the filename of the dynamic linker inside the respective *`libsubdir`*.

3. On the target system, invoke the dynamic linker with your executable as:

```
> sysroot/libsubdir/ldso \
  --library-path sysroot/libsubdir:sysroot/usr/libsubdir \
  /path/to/your-executable
```

   For the default multilib, this becomes:

```
> sysroot/lib/ld.so.1 \
  --library-path sysroot/lib:sysroot/usr/lib \
  /path/to/your-executable
```

Invoking the linker in this manner requires that you provide either an absolute pathname to your executable, or a relative pathname prefixed with `./`. Specifying only the name of a file in the current directory does not work.

3.6. Using GDB Server for Debugging

The GDB server utility provided with Sourcery CodeBench Lite can be used to debug a GNU/Linux application. While Sourcery CodeBench runs on your host system, `gdbserver` and the target application run on your target system. Even though Sourcery CodeBench and your application run on different systems, the debugging experience when using `gdbserver` is very similar to debugging a native application.

3.6.1. Running GDB Server

The GDB server executables are included in the sysroot in ABI-specific subdirectories of *`sysroot`*`/usr`. Use the executable from the sysroot and library subdirectory that match your program. See Section 3.2, “Library Configurations” for details.

You must copy the sysroot to your target system as described in Section 3.5.1, “Installing the Sysroot”. You must also copy the executable you want to debug to your target system.

If you have installed the sysroot in the root directory of the filesystem on the target, you can invoke `gdbserver` as:

```
> gdbserver :10000 program arg1 arg2 ...
```

where *`program`* is the path to the program you want to debug and *`arg1 arg2 ...`* are the arguments you want to pass to it. The `:10000` argument indicates that `gdbserver` should listen for connections from GDB on port 10000. You can use a different port, if you prefer.

If you have installed the sysroot in an alternate directory, invoking `gdbserver` becomes more complicated. You must build your application using the link-time options to specify the location of the sysroot, as described in Section 3.5.2, “Using Linker Options to Specify the Sysroot Location”. You must also invoke `gdbserver` itself using the dynamic linker provided in the Sourcery CodeBench sysroot, as described in Section 3.5.3, “Specifying the Sysroot Location at Runtime”. In other words, the command to invoke `gdbserver` in this case would be similar to:

```
> sysroot/libsubdir/ldso \
  --library-path sysroot/libsubdir:sysroot/usr/libsubdir \
  sysroot/usr/libsubdir/bin/gdbserver :10000 \
  program arg1 arg2 ...
```

For the default multilib, this becomes:

```
> sysroot/lib/ld.so.1 \
  --library-path sysroot/lib:sysroot/usr/lib \
  sysroot/usr/lib/bin/gdbserver :10000 \
  program arg1 arg2 ...
```

3.6.2. Connecting to GDB Server from the Debugger

You can connect to GDB server by using the following command from within GDB:

```
(gdb) target remote target:10000
```

where *target* is the host name or IP address of your target system.

When your program exits, `gdbserver` exits too. If you want to debug the program again, you must restart `gdbserver` on the target. Then, in GDB, reissue the `target` command shown above.

3.6.3. Setting the Sysroot in the Debugger

In order to debug shared libraries, GDB needs to map the pathnames of shared libraries on the target to the pathnames of equivalent files on the host system. Debugging of multi-threaded applications also depends on correctly locating copies of the libraries provided in the sysroot on the host system.

In some situations, the target pathnames are valid on the host system. In particular, this is the case if you are running GDB on a Linux host and using QEMU user-space emulation on that same host as your target. Otherwise, you must tell GDB how to map target pathnames onto the equivalent host pathnames.

In the general case, there are two GDB commands required to set up the mapping:

```
(gdb) set sysroot-on-target target-pathname
(gdb) set sysroot host-pathname
```

This causes GDB to replace all instances of the *target-pathname* prefix in shared library pathnames reported by the target with *host-pathname* to get the location of the equivalent library on the host.

If you have installed the sysroot in the root filesystem on the target, you can omit the `set sysroot-on-target` command, and use only `set sysroot` to specify the location on the host system.

Refer to Section 3.5.1, “Installing the Sysroot” for more information about installing the sysroot on the target. Note that if you have installed a stripped copy of the provided libraries on the target, you should give GDB the location of an unstripped copy on the host.

3.7. GENIVI 3.0 Compliance

GENIVI is a non-profit industry alliance of automotive OEMs working on adoption of an In-Vehicle Infotainment (IVI) open-source development platform. Mentor Graphics is a member of the alliance.

A change made to the Mentor Embedded Linux kernel to support the GENIVI 3.0 specification required that same support be added to the GNU C Library included in Sourcery CodeBench.

The new AF_BUS socket address family, used in kernel inter-process communication, is included in GLIBC for Sourcery CodeBench Lite..

This change has not been incorporated in the upstream Linux kernel sources nor into upstream GLIBC. Therefore, this release of Sourcery CodeBench may not be compatible with future versions of the Linux kernel.

For more information about GENIVI, see the alliance website[1].

3.8. Using OpenMP

Sourcery CodeBench Lite for MIPS GNU/Linux includes the GNU OpenMP library (libgomp). This is an API that supports multi-platform shared-memory parallel programming.

To compile programs that use OpenMP features, use the `-fopenmp` command-line option. For more information about OpenMP, see `http://www.openmp.org/`.

[1] http://www.genivi.org

Chapter 4
Using Sourcery CodeBench from the Command Line

This chapter demonstrates the use of Sourcery CodeBench Lite from the command line.

4.1. Building an Application

This chapter explains how to build an application with Sourcery CodeBench Lite using the command line. As elsewhere in this manual, this section assumes that your target system is mips-linux-gnu, as indicated by the `mips-linux-gnu` command prefix.

Using an editor (such as `notepad` on Microsoft Windows or `vi` on UNIX-like systems), create a file named `main.c` containing the following simple factorial program:

```
#include <stdio.h>

int factorial(int n) {
  if (n == 0)
    return 1;
  return n * factorial (n - 1);
}

int main () {
  int i;
  int n;
  for (i = 0; i < 10; ++i) {
    n = factorial (i);
    printf ("factorial(%d) = %d\n", i, n);
  }
  return 0;
}
```

Compile and link this program using the command:

```
> mips-linux-gnu-gcc -o factorial main.c
```

There should be no output from the compiler. (If you are building a C++ application, instead of a C application, replace `mips-linux-gnu-gcc` with `mips-linux-gnu-g++`.)

4.2. Running Applications on the Target System

You may need to install the Sourcery CodeBench runtime libraries and dynamic linker on the target system before you can run your application. Refer to Chapter 3, “Sourcery CodeBench Lite for MIPS GNU/Linux” for specific instructions.

To run your program on a GNU/Linux target system, use the command:

```
> factorial
```

You should see:

```
factorial(0) = 1
factorial(1) = 1
factorial(2) = 2
factorial(3) = 6
factorial(4) = 24
factorial(5) = 120
factorial(6) = 720
factorial(7) = 5040
```

```
factorial(8) = 40320
factorial(9) = 362880
```

4.3. Running Applications with QEMU

On Linux hosts, Sourcery CodeBench Lite includes the QEMU user-space emulator. This is a program which runs on your host computer and allows you to run and debug MIPS GNU/Linux applications without target hardware. Because QEMU's user-space emulation works by translating Linux system calls made by the target program into corresponding calls on the host Linux system, you cannot run this emulator on a Windows host.

To use QEMU, you first need to find the sysroot directory in your Sourcery CodeBench Lite installation that matches your compilation options, as described in Section 3.5, "Using Sourcery CodeBench Lite on GNU/Linux Targets". For example, the sysroot for the default multilib is in the `mips-linux-gnu/libc` subdirectory of your install directory.

Applications run with the QEMU emulator share the environment and file system of the host computer, so you do not need to copy the sysroot. However, you must tell QEMU where to find the sysroot on the host. You can do this by providing appropriate linker options when building your application, as described in Section 3.5.2, "Using Linker Options to Specify the Sysroot Location". Alternatively, you can invoke the dynamic linker directly to run your program, as described in Section 3.5.3, "Specifying the Sysroot Location at Runtime".

If you use linker options to set the sysroot pathname, you can invoke QEMU as:

```
> mips-linux-gnu-qemu program arguments...
```

where *`program`* is the executable to run, and *`arguments`* are any command-line arguments to your program.

If you want to invoke the dynamic linker directly instead, use a command similar to:

```
> mips-linux-gnu-qemu sysroot/lib/ld.so.1 \
  --library-path sysroot/lib:sysroot/usr/lib \
  program arguments...
```

Refer to Section 3.5.3, "Specifying the Sysroot Location at Runtime" for additional details on running the dynamic linker.

You can specify emulation of a specific CPU by providing the `--cpu` option to QEMU. The default is `24Kf`. Additional supported CPU emulations include `4Kc`, `4Km`, `4KEcR1`, `4KEmR1`, `4KEc`, `4KEm`, `24Kc`, `34Kf`, `M14K`, and `M14Kc`.

By default, QEMU reports the minimum Linux kernel version required by the GNU C library included with this release of Sourcery CodeBench to applications via the `uname` system call. You can specify some other kernel version using the `-r` command-line option. QEMU's emulation is generally independent of the kernel version on the host Linux system.

QEMU includes a built-in GDB server which you can use to debug your program with GDB. Start QEMU with the `-g` *`port`* command-line option. You can generally choose any port number that is not already in use on your host system. Then, follow the instructions in Section 4.4.2, "Connecting to an External GDB Server" to connect from GDB.

The version of QEMU included with Sourcery CodeBench Lite for MIPS GNU/Linux is configured to run in user-space emulation mode only, and other QEMU features not documented here are not

supported in Sourcery CodeBench Lite. For additional information about QEMU, visit the QEMU web site[1].

4.4. Running Applications from GDB

You can run GDB, the GNU Debugger, on your host system to debug programs running remotely on a target board or system.

When starting GDB, give it the pathname to the program you want to debug as a command-line argument. For example, if you have built the factorial program as described in Section 4.1, “Building an Application”, enter:

```
> mips-linux-gnu-gdb factorial
```

While this section explains the alternatives for using GDB to run and debug application programs, explaining the use of the GDB command-line interface is beyond the scope of this document. Please refer to the GDB manual for further instructions.

4.4.1. Connecting to the Sourcery CodeBench Debug Sprite

The Sourcery CodeBench Debug Sprite is a program that runs on the host system to support hardware debugging devices. You can use the Debug Sprite to run and debug programs on a target board without an operating system, or to debug an operating system kernel. See Chapter 5, “Sourcery CodeBench Debug Sprite” for detailed information about the supported devices.

You can start the Sprite directly from within GDB:

```
(gdb) target remote | mips-linux-gnu-sprite arguments
```

Refer to Section 5.2, “Invoking Sourcery CodeBench Debug Sprite” for a full description of the Sprite arguments.

4.4.2. Connecting to an External GDB Server

Sourcery CodeBench Lite includes a program called `gdbserver` that can be used to debug a program running on a remote MIPS GNU/Linux target. Follow the instructions in Chapter 3, “Sourcery CodeBench Lite for MIPS GNU/Linux” to install and run `gdbserver` on your target system.

From within GDB, you can connect to a running `gdbserver` or other debugging stub that uses the GDB remote protocol using:

```
(gdb) target remote host:port
```

where *`host`* is the host name or IP address of the machine the stub is running on, and *`port`* is the port number it is listening on for TCP connections.

You can also use this method to connect to a QEMU user-space emulator that you have started with the `-g` *`port`* option. In the usual case where QEMU is running on the same host as the debugger, you can omit the *`host`* part of the GDB command and simply use:

```
(gdb) target remote :port
```

For more information about starting QEMU, see Section 4.3, “Running Applications with QEMU”.

[1] http://fabrice.bellard.free.fr/qemu

4.5. Using the Prelinker

You can make programs that use ELF shared libraries start up faster by processing the binaries with the prelinker. This utility is called the *prelinker* because it preprocesses the application binaries and shared libraries before dynamic linking. On the other hand, prelinking is performed *after* the normal linking step (`mips-linux-gnu-ld` invocation) that builds the executables and shared libraries.

Prelinking operates by performing some of the functions of the dynamic linker in advance. Specifically, it pre-assigns ELF shared libraries to virtual address slots that are the same in all prelinked applications, and writes the modified relocation information back to the ELF files for the affected libraries and applications. At program startup time, the dynamic linker checks whether the precomputed shared library relocations are still valid; if they are, several initialization steps can be skipped, resulting in an improved startup time. If not (for example, if a shared library has been modified since the program was prelinked), it falls back on the normal dynamic linking process. Prelinking is reversible and can be re-run to perform incremental updates when applications or the libraries they depend on change.

Prelinking only works on shared libraries that are explicitly linked with application programs. If your application uses `dlopen` to load shared libraries, that operation is not affected by the prelinker. This is because the prelinker cannot predict in advance which libraries are to be loaded with `dlopen`, or in what order.

For best results, the prelinker should be run on an entire set of executables and their dependent libraries at once, rather than on application executables individually. For example, the native prelinker shipped with many Linux systems is intended to operate on all of the standard system utilities and libraries. The version of the prelinker included in Sourcery CodeBench Lite additionally supports prelinking of a cross-built root file system. Invoke `mips-linux-gnu-prelink` with the `--root=`*`pathname`* command-line option to specify the path to the root of the target file system. All other pathnames referenced by the prelinker, including those of any executables specified on the command line, are relative to this root.

4.5.1. Configuration File

The first step in running the prelinker is to create a configuration file. This is canonically called `/etc/prelink.conf` in the target file system. Even if you plan to only prelink selected application binaries, you should create a configuration file listing the directories to search for the shared libraries referenced by the applications. The prelinker does not prelink against libraries that are neither findable from the configuration file, nor explicitly listed on the command line.

The contents of the configuration file are a list of directories on the target file system to recursively scan, and patterns matching names of files to be skipped. Specifically, it can contain lines of the following forms:

- Comment lines starting with #.
- Directory names to be recursively searched. Optionally, directory names can be prefixed with options `-l` (do not search across file systems) or `-h` (follow symbolic links).
- Blacklist patterns prefixed with the `-b` option, which specify directories and files to be skipped during the recursive search. The operand can be either a directory name, the name of an individual file, or a glob pattern to be matched against filenames in each directory.

For example, here is a simple configuration file.

```
# Search in these directories.
-l /lib
-l /usr/lib
# Ignore files matching these patterns.
-b /lib/firmware
-b *.h
-b *.xml
-b *.xslt
```

4.5.2. Invoking the Prelinker

The prelinker has two primary modes of operation:

1. Prelink only certain executables against a set of libraries.

   ```
   > mips-linux-gnu-prelink --root=pathname executable1 ...
   ```

   In this case, the executables to be prelinked are specified on the command line. The configuration file is used only to search for the shared libraries that are used by the given programs. If you wish, you can also specify additional library pathnames on the command line.

2. Prelink all the libraries and executables in a given set of directories.

   ```
   > mips-linux-gnu-prelink -a --root=pathname
   ```

   If the `-a` option is specified, then the directories listed in the configuration file are searched for executables to be prelinked, as well as for the libraries they reference. While this usage also permits additional executables and libraries to be specified on the command line, the recommended practice is to list all files of interest in the configuration file instead.

Refer to the man page for the prelinker for documentation for other command-line options.

Chapter 5
Sourcery CodeBench Debug Sprite

This chapter describes the use of the Sourcery CodeBench Debug Sprite for remote debugging. The Sprite is provided for debugging of the Linux kernel on the target board. This chapter includes information about the debugging devices and boards supported by the Sprite for MIPS GNU/Linux.

Sourcery CodeBench Lite contains the Sourcery CodeBench Debug Sprite for MIPS GNU/Linux. This Sprite is provided to allow debugging of programs running on a bare board. You can use the Sprite to debug a program when there is no operating system on the board, or for debugging the operating system itself. If the board is running an operating system, and you wish to debug a program running on that OS, you should use the facilities provided by the OS itself (for instance, using `gdbserver`).

The Sprite acts as an interface between GDB and external debug devices and libraries. Refer to Section 5.2, "Invoking Sourcery CodeBench Debug Sprite" for information about the specific devices supported by this version of Sourcery CodeBench Lite.

Note for Linux users

The Debug Sprite provided with Sourcery CodeBench Lite allows remote debugging of the Linux kernel running on the target. For remote debugging of application programs, you should use `gdbserver` instead. See Chapter 3, "Sourcery CodeBench Lite for MIPS GNU/Linux" for details about how to install and run `gdbserver` on the target.

Important

The Sourcery CodeBench Debug Sprite is not part of the GNU Debugger and is not free or open-source software. You may use the Sourcery CodeBench Debug Sprite only with the GNU Debugger. You may not distribute the Sourcery CodeBench Debug Sprite to any third party.

5.1. Probing for Debug Devices

Before running the Sourcery CodeBench Debug Sprite for the first time, or when attaching new debug devices to your host system, it is helpful to verify that the Sourcery CodeBench Debug Sprite recognizes your debug hardware. From the command line, invoke the Sprite with the `-i` option:

```
> mips-linux-gnu-sprite -i
```

This prints out a list of supported device types. For devices that can be autodetected, it additionally probes for and prints out a list of attached devices. For instance:

```
Sourcery CodeBench Debug Sprite for MIPS (Sourcery CodeBench Lite \
2013.05-66)
mdi: [lib=<file>&cfg=<file>&rst=<n>] MDI device
  mdi:///23/1 - 24KE     (Instruction)/24KE LE
  mdi:///23/2 - 24KE     (Instruction)/24KE BE
  mdi:///24/1 - 24KE     (Cycle)/24KE LE
  mdi:///24/2 - 24KE     (Cycle)/24KE BE
  mdi:///$Target/$Device - Generic MDI target/device
```

This shows that MDI (Microprocessor Debug Interface) devices are supported. Four MIPSsim devices have been autodetected. Note that additional configuration steps for the MDI library are required to allow the Sprite to autodetect devices; see Section 5.4, "MDI Devices".

5.2. Invoking Sourcery CodeBench Debug Sprite

The Debug Sprite is invoked as follows:

```
> mips-linux-gnu-sprite [options] device-url board-file
```

The *device-url* specifies the debug device to use to communicate with the board. It follows the standard format:

```
scheme:scheme-specific-part[?device-options]
```

Most device URL schemes also follow the regular format:

```
scheme:[//hostname:[port]]/path[?device-options]
```

The meanings of *hostname*, *port*, *path* and *device-options* parts depend on the *scheme* and are described below. The following schemes are supported in Sourcery CodeBench Lite for MIPS GNU/Linux:

`mdi` Use a Microprocessor Debug Interface (MDI) debugging device. Refer to Section 5.4, “MDI Devices”.

The optional `?`*device-options* portion is allowed in all schemes. These allow additional device-specific options of the form *name=value*. Multiple options are concatenated using `&`.

The *board-file* specifies an XML file that describes how to initialize the target board, as well as other properties of the board used by the debugger. If *board-file* refers to a file (via a relative or absolute pathname), it is read. Otherwise, *board-file* can be a board name, and the toolchain's board directory is searched for a matching file. See Section 5.6, “Supported Board Files” for the list of supported boards, or invoke the Sprite with the `-b` option to list the available board files. You can also write a custom board file; see Section 5.7, “Board File Syntax” for more information about the file format.

Both the *device-url* and *board-file* command-line arguments are required to correctly connect the Sprite to a target board.

5.3. Sourcery CodeBench Debug Sprite Options

The following command-line options are supported by the Sourcery CodeBench Debug Sprite:

`-a` Attach to a process already running on the target. Without this option, the default behavior is to reset the target on the initial connection, in preparation for loading a new program from the debugger.

`-b` Print a list of *board-file* files in the board config directory.

`-h` Print a list of options and their meanings. A list of *device-url* syntaxes is also shown.

`-i` Print a list of the accessible devices. If a *device-url* is also specified, only devices for that device type are scanned. Each supported device type is listed along with the options that can be appended to the *device-url*. For each discovered device, the *device-url* is printed along with a description of that device.

`-l` *[host]:port* Specify the host address and port number to listen for a GDB connection. If this option is not given, the Debug Sprite communicates with GDB using stdin and stdout. If you start the Sprite from within GDB using the `target remote | mips-linux-gnu-sprite ...` command, you do not need this option.

`-m` Listen for multiple sequential connections. Normally the Debug Sprite terminates after the first connection from GDB terminates. This option instead makes it listen for a subsequent connection. To terminate the Sprite, open a connection and send the string `END\n`.

`-q` Do not print any messages.

`-v` Print additional messages.

If any of `-b`, `-i` or `-h` are given, the Debug Sprite terminates after providing the information rather than waiting for a debugger connection.

5.4. MDI Devices

The Sourcery CodeBench Debug Sprite for MIPS supports MDI (Microprocessor Debug Interface) devices. Each MDI device is identified by a target number and device number; these form the *`path`* part of the device URL, and the *`hostname`* and *`port`* must be empty or omitted. Thus, the *`device-url`* has the form:

```
mdi:///targetnum/devicenum[?device-options]
```

You can also use the environment variables `GDBMDITARGET` and `GDBMDIDEVICE` to provide defaults for the *`targetnum`* and *`devicenum`*.

The following *`device-options`* are permitted:

`lib=`*`filename`* This option specifies the MDI library to load. It is equivalent to setting the `GDBMDILIB` environment variable.

`cfg=`*`filename`* Some MDI target libraries, such as MIPSsim, require a configuration file. (This is distinct from the Sprite's own *`board-file`*.) You can use this option to specify the file. It is equivalent to setting the `GDBMIPSSIMCONFIG` environment variable.

`rst=`*`seconds`* This option can be used to specify a delay after the target is reset by the Sprite. If the value of *`seconds`* is greater than zero, then execution is resumed for the specified number of seconds; this can be used to allow power-on firmware to initialize the memory controller and peripherals. Then the target is halted again and queried for configuration.

If the value of *`seconds`* is `-1`, then the target is queried immediately without reset. This is the same effect as passing the `-a` command-line option to the Sprite, which allows the Sprite to attach to a running program.

This option is equivalent to setting the `GDBMDICONNRST` environment variable. If neither the option nor the environment variable are provided, the default is to reset the target and query it immediately unless the `-a` option is specified.

`group=/targetn/devicen` This option may be specified multiple times and is cumulative. Each of the specified devices is opened and queried and they are all treated as threads of execution, subject to being enabled or active; if a device is disabled or has no active thread contexts

associated with it, it is not visible to GDB but is still under control of the Sprite in case its state changes. This option cannot be used in combination with the `team=` option.

`team=/targetn/devicen` This option may be specified multiple times and is cumulative. The specified devices are not opened, but are associated with the base device by means of the MDI team mechanism for the purpose of synchronization. The specified devices may still be opened and controlled by another debugger (such as another instance of the Debug Sprite) independently. This option cannot be used in combination with the `group=` option.

Before you can connect to a target using the MDI API, you must tell the Debug Sprite which shared library or DLL to load for your simulator or device. On Linux hosts you should add the directory containing the shared library files to your `LD_LIBRARY_PATH` environment variable. On Windows hosts, add the directory containing the DLLs to your `PATH` environment variable. Then, either set the environment variable `GDBMDILIB` to the base name of the MDI library before starting the Debug Sprite, or use the `lib=` device option to specify the library to load.

Similarly, the `-i` command-line option can only probe for devices if you have set the `PATH` or `LD_LIBRARY_PATH` environment variable appropriately, and specify an MDI library using either the `GDBMDILIB` environment variable or the `lib=` device option. Otherwise, it reports only the generic *`device-url`* syntax.

For example, to use an FS2 probe on a Windows host to debug a MIPS Malta board, first add the directory containing the MDI DLLs to your `PATH`. Then you can invoke the Sprite from GDB using a command line similar to:

```
(gdb) target remote | mips-linux-gnu-sprite \
'mdi:///2/2?lib=jnetfs2mdilib.dll&rst=7' malta
```

The quotes are required to prevent special characters in the *`device-url`* from being interpreted by the shell.

In the above command, the `rst=7` option provides for a sufficient delay for the board's reset code to execute on connection. Since this takes several seconds, GDB may time out waiting for the Sprite to respond. You can prevent this by issuing this command before you connect to the Sprite:

```
(gdb) set remotetimeout 10
```

To use the Sprite with MIPSsim, a configuration file is required. The configuration files provided with the MIPSsim distribution are intended for use with standalone execution from the command line, rather than running the program from the debugger. So, make a copy and comment out the `APP_FILE` setting. It is also recommended that you comment out `TRACE_FILE` as well, since the trace files can be very large.

To connect to MIPSsim using the Sprite on a Linux host, first set your `LD_LIBRARY_PATH` and `GDBMDILIB` as described above. You can run the Sprite from the shell to probe for devices to verify that your setup is correct:

```
> mips-linux-gnu-sprite -i
```

Then, from GDB, use a command similar to:

```
(gdb) target remote | mips-linux-gnu-sprite \
'mdi:///23/2?cfg=24KE.cfg&rst=-1' mipssim
```

Fill in your target and device numbers as reported by the probe output, and the full pathname to your configuration file. The `rst=-1` option is required, as MIPSsim does not support reset.

This section describes only the basic MDI usage; refer to the documentation for your MDI simulator or debug device for details specific to that target. Note, in particular, that some MDI targets may require you to set up a license in addition to the steps given here.

5.5. Debugging a Remote Board

You can run the Sourcery CodeBench Debug Sprite on a different machine from the one on which GDB is running. For example, if your board is connected to a machine in your lab, you can run the debugger on your laptop and connect to the remote board. The Sourcery CodeBench Debug Sprite must run on the machine that is connected to the target board. You must have Sourcery CodeBench installed on both machines.

To use this mode, you must start the Sprite with the `-l` option and specify the port on which you want it to listen. For example:

```
> mips-linux-gnu-sprite -l :10000 device-url board-file
```

starts the Sprite listening on port 10000.

When running GDB from the command line, use the following command to connect GDB to the remote Sprite:

```
(gdb) target remote host:10000
```

where *`host`* is the name of the remote machine. After this, debugging is just as if you are debugging a target board connected to your host machine.

For more detailed instructions on using the Sourcery CodeBench Debug Sprite in this way, please refer to the Sourcery CodeBench Knowledge Base[1].

5.6. Supported Board Files

The Sourcery CodeBench Debug Sprite for MIPS GNU/Linux includes support for the following target boards. Specify the appropriate *`board-file`* as an argument when invoking the Sprite from the command line.

Board	**Config**
MIPS Malta	`malta`
MIPS Malta 64-bit	`malta64`
MIPS SEAD-3 LX110	`sead3-lx110`
MIPS SEAD-3 LX50	`sead3-lx50`
MIPSsim	`mipssim`

[1] https://sourcery.mentor.com/GNUToolchain/kbentry132

5.7. Board File Syntax

The *board-file* can be a user-written XML file to describe a non-standard board. The Sourcery CodeBench Debug Sprite searches for board files in the `mips-linux-gnu/lib/boards` directory in the installation. Refer to the files in that directory for examples.

The file's DTD is:

```
<!-- Board description files

     Copyright (c) 2007-2012 Mentor Graphics Corporation.

     THIS FILE CONTAINS PROPRIETARY, CONFIDENTIAL, AND TRADE
     SECRET INFORMATION OF MENTOR GRAPHICS AND/OR ITS LICENSORS.

     You may not use or distribute this file without the express
     written permission of Mentor Graphics or its authorized
     distributor.  This file is licensed only for use with
     Sourcery CodeBench.  No other use is permitted.
     -->

<!ELEMENT board
 (category?, properties?, feature?, initialize?, memory-map?, \
debuggerDefaults?)>

<!-- Board category to group boards list into the tree -->
<!ELEMENT category (#PCDATA)>

<!ELEMENT properties
 (description?, property*)>

<!ELEMENT initialize
 (write-register | write-memory | delay
  | wait-until-memory-equal | wait-until-memory-not-equal)* >
<!ELEMENT write-register EMPTY>
<!ATTLIST write-register
          address CDATA   #REQUIRED
                        value   CDATA   #REQUIRED
                        bits    CDATA   #IMPLIED>
<!ELEMENT write-memory EMPTY>
<!ATTLIST write-memory
          address CDATA   #REQUIRED
                        value   CDATA   #REQUIRED
                        bits    CDATA   #IMPLIED>
<!ELEMENT delay EMPTY>
<!ATTLIST delay
          time CDATA   #REQUIRED>
<!ELEMENT wait-until-memory-equal EMPTY>
<!ATTLIST wait-until-memory-equal
          address CDATA   #REQUIRED
                        value   CDATA   #REQUIRED
                        timeout CDATA   #IMPLIED
                        bits    CDATA   #IMPLIED>
```

```
<!ELEMENT wait-until-memory-not-equal EMPTY>
<!ATTLIST wait-until-memory-not-equal
          address CDATA   #REQUIRED
                        value   CDATA   #REQUIRED
                        timeout CDATA   #IMPLIED
                        bits    CDATA   #IMPLIED>

<!ELEMENT memory-map (memory-device)*>
<!ELEMENT memory-device (property*, description?, sectors*)>
<!ATTLIST memory-device
                        address CDATA   #REQUIRED
          size    CDATA   #REQUIRED
          type    CDATA   #REQUIRED
                        device  CDATA   #IMPLIED>

<!ELEMENT description (#PCDATA)>
<!ELEMENT property (#PCDATA)>
<!ATTLIST property name CDATA #REQUIRED>
<!ELEMENT sectors EMPTY>
<!ATTLIST sectors
 size CDATA #REQUIRED
 count CDATA #REQUIRED>

<!-- Definition of default option values for each debug interface -->
<!ELEMENT debuggerDefaults (debugInterface*)>
<!ELEMENT debugInterface (option*)>
<!ATTLIST debugInterface
 name CDATA #REQUIRED
>
<!ELEMENT option EMPTY>
<!ATTLIST option
 name CDATA #REQUIRED
 defaultValue CDATA #REQUIRED
>

<!ENTITY % gdbtarget SYSTEM "gdb-target.dtd">
%gdbtarget;
```

All values can be provided in decimal, hex (with a `0x` prefix) or octal (with a `0` prefix). Addresses and memory sizes can use a `K`, `KB`, `M`, `MB`, `G` or `GB` suffix to denote a unit of memory. Times must use a `ms` or `us` suffix.

The following elements are available:

`<board>` — This top-level element encapsulates the entire description of the board. It can contain `<category>`, `<properties>`, `<feature>`, `<initialize>` and `<memory-map>` elements.

`<category>` — The `<category>` element specifies a '.' separated categorization of this board (e.g., Vendor.Family) to allow grouping similar boards in a tree structure.

`<properties>`	The `<properties>` element specifies specific properties of the target system. This element can occur at most once. It can contain a `<description>` element.
`<initialize>`	The `<initialize>` element defines an initialization sequence for the board, which the Sprite performs before downloading a program. It can contain `<write-register>`, `<write-memory>` and `<delay>` elements.
`<feature>`	This element is used to inform GDB about additional registers and peripherals available on the board. It is passed directly to GDB; see the GDB manual for further details.
`<memory-map>`	This element describes the memory map of the target board. It is used by GDB to determine where software breakpoints may be used and when flash programming sequences must be used. This element can occur at most once. It can contain `<memory-device>` elements.
`<memory-device>`	This element specifies a region of memory. It has four attributes: `address`, `size`, `type` and `device`. The `address` and `size` attributes specify the location of the memory device. The `type` attribute specifies that device as `ram`, `rom` or `flash`. The `device` attribute is required for `flash` regions; it specifies the flash device type. The `<memory-device>` element can contain a `<description>` element.
`<write-register>`	This element writes a value to a control register. It has three attributes: `address`, `value` and `bits`. The `bits` attribute, specifying the bit width of the write operation, is optional; it defaults to 32.
`<write-memory>`	This element writes a value to a memory location. It has three attributes: `address`, `value` and `bits`. The `bits` attribute is optional and defaults to 32. Bit widths of 8, 16 and 32 bits are supported. The address written to must be naturally aligned for the size of the write being done.
`<delay>`	This element introduces a delay. It has one attribute, `time`, which specifies the number of milliseconds, or microseconds to delay by.
`<description>`	This element encapsulates a human-readable description of its enclosing element.
`<property>`	The `<property>` element allows additional name/value pairs to be specified. The property name is specified in a `name` attribute. The property value is the body of the `<property>` element.
`<debuggerDefaults>`	The `<debuggerDefaults>` element defines the default option values for each debug interface.

Chapter 6
Next Steps with Sourcery CodeBench

This chapter describes where you can find additional documentation and information about using Sourcery CodeBench Lite and its components.

6.1. Sourcery CodeBench Knowledge Base

The Sourcery CodeBench Knowledge Base is available to registered users at the Sourcery CodeBench Portal[1]. Here you can find solutions to common problems including installing Sourcery CodeBench, making it work with specific targets, and interoperability with third-party libraries. There are also additional example programs and tips for making the most effective use of the toolchain and for solving problems commonly encountered during debugging. The Knowledge Base is updated frequently with additional entries based on inquiries and feedback from customers.

6.2. Manuals for GNU Toolchain Components

Sourcery CodeBench Lite includes the full user manuals for each of the GNU toolchain components, such as the compiler, linker, assembler, and debugger. Most of the manuals include tutorial material for new users as well as serving as a complete reference for command-line options, supported extensions, and the like.

When you install Sourcery CodeBench Lite, links to both the PDF and HTML versions of the manuals are created in the shortcuts folder you select. If you elected not to create shortcuts when installing Sourcery CodeBench Lite, the documentation can be found in the `share/doc/mips-mips-linux-gnu/` subdirectory of your installation directory.

In addition to the detailed reference manuals, Sourcery CodeBench Lite includes a Unix-style manual page for each toolchain component. You can view these by invoking the `man` command with the pathname of the file you want to view. For example, you can first go to the directory containing the man pages:

```
> cd $INSTALL/share/doc/mips-mips-linux-gnu/man/man1
```

Then you can invoke `man` as:

```
> man ./mips-linux-gnu-gcc.1
```

Alternatively, if you use `man` regularly, you'll probably find it more convenient to add the directory containing the Sourcery CodeBench man pages to your `MANPATH` environment variable. This should go in your `.profile` or equivalent shell startup file; see Section 2.6, “Setting up the Environment” for instructions. Then you can invoke `man` with just the command name rather than a pathname.

Finally, note that every command-line utility program included with Sourcery CodeBench Lite can be invoked with a `--help` option. This prints a brief description of the arguments and options to the program and exits without doing further processing.

[1] https://sourcery.mentor.com/GNUToolchain/

Appendix A
Sourcery CodeBench Lite Release Notes

This appendix contains information about changes in this release of Sourcery CodeBench Lite for MIPS GNU/Linux. You should read through these notes to learn about new features and bug fixes.

A.1. Changes in Sourcery CodeBench Lite for MIPS GNU/Linux

This section documents Sourcery CodeBench Lite changes for each released revision.

A.1.1. Changes in Sourcery CodeBench Lite 2013.05-66

QEMU emulator. Sourcery CodeBench Lite now includes the QEMU user-mode emulator for Linux hosts. QEMU uses just-in-time translation of target machine instructions to provide high-performance simulation of target applications. System calls on the MIPS GNU/Linux target are translated into corresponding calls on the host Linux system. Refer to Chapter 4, “Using Sourcery CodeBench from the Command Line” for more information about using QEMU.

Prelinker. The prelinker package is now included in Sourcery CodeBench Lite for MIPS GNU/Linux.

OpenMP support. Support for the OpenMP application programming interface is now available in Sourcery CodeBench Lite for MIPS GNU/Linux. To compile programs that use OpenMP features, use the `-fopenmp` command-line option. For more information about using OpenMP with Sourcery CodeBench Lite, see Section 3.8, “Using OpenMP”. For more information about the OpenMP API, see `http://www.openmp.org/`.

A.1.2. Changes in Sourcery CodeBench Lite 2013.05-36

C++ exception handling format. The C++ exception handling format has been changed to fix a defect in the specification. The new format is not backwards-compatible with code produced by Sourcery CodeBench beginning with 2012.09 releases; you should recompile all C++ code using exceptions. The compatibility issue does not affect code produced by older Sourcery CodeBench Lite releases or third-party libraries built with other compilers.

Exception handling bug fix. A bug that sometimes caused an incorrect call to terminate during exception handling has been fixed.

Temporary register assembler bug fix. A bug has been fixed that caused `$zero` to be used as a temporary register for address calculation on some loads targeting that register.

Assembler bug fix for `.stab` directives. An assembler bug has been fixed that caused incorrect code to be generated when `.stab` directives were used in microMIPS or MIPS16 mode.

Assembler bug fix for microMIPS symbol loads. An assembler bug has been fixed that caused some microMIPS symbols to be loaded from an incorrect address.

N64 linker assertion failure fix. A linker bug has been fixed that caused an assertion failure when linking N64 code using the special `__ehdr_start` symbol.

O32 PIC call assembler bug fix. A bug that caused incorrect code generation on O32 PIC function calls using the `R_MIPS_JALR` relocation has been fixed.

IEEE 754 soft float NaN interpretation bug fixes. The IEEE 754 soft-float emulation libraries have been corrected to consistently use the legacy NaN representations.

GLIBC NaN string conversion bug fix. A bug in GLIBC's `strtof`, `strtod`, and `strtold` functions has been fixed that caused them to incorrectly return a signaling NaN rather than a quiet

NaN given input of the form `"nan(N)"`. The bug also affected the `nanf`, `nan`, and `nanl` functions given a non-empty input string.

`gdbserver` hardware watchpoints fix. A `gdbserver` bug has been fixed that caused insertion of hardware watchpoints to fail.

Debug Sprite spurious error message bug fix. A bug in the Sourcery CodeBench Debug Sprite has been fixed that caused a spurious `Error initializing the target` message upon startup in the attachment (`-a`) mode.

A.1.3. Changes in Sourcery CodeBench Lite 2013.05-7

Conditional expression bug fix. A bug that caused an internal compiler error in some programs with conditional expressions has been fixed.

Missed optimization. A bug that caused GCC to miss an opportunity to use short instructions when compiling with `-mmicromips` has been fixed.

Exception handling bug fix. A bug that sometimes caused incorrect results when using C++ exception specifications has been fixed.

Incorrect optimization bug fix. A compiler bug has been fixed that caused incorrect code to be generated for some comparisons unless optimization was suppressed with `-fno-forward-propagate`.

Exception handling bug fix. A bug that caused unexpected runtime aborts when using C++ exception handling with `-muclibc` has been fixed.

GCC version 4.7.3. Sourcery CodeBench Lite for MIPS GNU/Linux is now based on GCC version 4.7.3. This update incorporates numerous bug fixes. For more information, see `http://gcc.gnu.org/gcc-4.7/changes.html`.

GCC option `-meva`. GCC now passes the `-meva` command-line option to GAS.

IEEE 754-2008 features and multilibs added. Support for some IEEE 754-2008 floating-point arithmetic standard features has been added to the toolchain for processors that implement them. This includes changes to the compiler, binutils, the dynamic linker and floating-point environment library functions. The changes enable support for the recommended encoding of NaN data introduced by the said revision of the standard as well as the use of the floating-point `ABS.fmt` and `NEG.fmt` instructions in a non-arithmetic manner. Multilibs built with the updated encoding of NaN data have been added. For information as to how to control these features, please refer to the compiler and assembler manuals.

Linker `--gc-sections` crash fix. A bug that caused the linker to sometimes crash when using `--gc-sections` has been fixed.

Improved error message for incompatible object files. The linker error message emitted when attempting to link objects with incompatible error handling formats now includes the object names.

Binutils update. The binutils package has been updated to version 2.23.52.20130219 from the FSF trunk. This update includes numerous bug fixes.

Installer warnings fixed. A bug that caused Gtk warnings relating to `libappmenu.so` when running the installer on 64-bit Ubuntu GNU/Linux hosts has been fixed.

Fix for installer upgrade problems. Sourcery CodeBench Lite for MIPS GNU/Linux can now be installed into a directory already containing a previous version.

EGLIBC version 2.17. Sourcery CodeBench Lite for MIPS GNU/Linux now includes EGLIBC version 2.17 library which is based on GNU C Library version 2.17. For more information about changes, see `http://www.eglibc.org/news#eglibc_2_17`. This version requires that Linux kernel version 2.6.16 or later be installed on the target in order to run applications.

New GLIBC macro `issignaling`. A new `<math.h>` macro named `issignaling` to check for a signaling Not a Number (sNaN) has been added to GLIBC. Please see the manual for further information.

GLIBC IEEE 754-2008 standards conformance. Math functions provided by GLIBC's `libm` have been fixed to return a quiet Not a Number (qNaN) when they wrongly returned a signaling Not a Number (sNaN).

Linux kernel headers update. Linux kernel header files have been updated to version 3.8.2.

Improved source line stepping. GDB and `gdbserver` now implement range stepping, which improves the performance of single stepping over a source line by reducing the number of control messages from GDB.

GDB hang fix. A bug that caused GDB to sometimes hang when setting a breakpoint has been fixed.

Sprite remote communication bug fix. A bug in the Sourcery CodeBench Debug Sprite that caused it to exit with the error `Remote communication error. Target disconnected.: Invalid argument.` on some Microsoft Windows hosts has been fixed.

Debug Sprite with multiple connections. The Sourcery CodeBench Debug Sprite now works correctly when using multiple connections (`-m`) with the Mentor Embedded Sourcery Probe. Previously, the Sprite exited with an error after the first connection.

Debug Sprite source line stepping improvement. The Sourcery CodeBench Debug Sprite now supports range stepping when used with debug devices that implement this MDI feature.

A.1.4. Changes in Sourcery CodeBench Lite 2012.09-99

PIC call optimization. A compiler bug that disabled PIC call optimizations has been fixed.

microMIPS branch generation fixed. A bug in GCC has been fixed that caused it to generate out-of-range microMIPS branches, which resulted in a linker error `relocation truncated to fit: R_MICROMIPS_PC16_S1`.

Compact C++ language-specific data. GCC now defaults to emitting a compact encoding of the C++ language-specific data for exception handling. For more information, refer to the documentation of the `-mcompact-eh` and `-mno-compact-eh` options in the GCC manual.

Pointer comparison bug fixed. A bug in GCC that caused it to incorrectly optimize away a pointer comparison has been fixed.

Loop optimization bug fix. A compiler bug that caused some forms of loop to be mis-optimized when using the `-fpromote-loop-indices` option has been fixed.

Wrong-code bug fix. A bug in GCC's scheduler has been fixed that sometimes caused incorrect code to be generated.

Performance regression fixed. A bug that introduced unnecessary instructions to zero-extend unsigned `char` or `short` values has been fixed.

Linker raw binary input crash fix. A bug that caused the linker to crash when linking binary inputs (`--format=binary`) while using `--gc-sections` has been fixed.

Assembler bug fixes. Assembler bugs have been fixed that caused assembly errors or incorrect code to be produced for the `LUI` instruction, microMIPS `B16`, `BEQZ16`, and `BNEZ16` suffixed instruction mnemonics, as well as 32-bit microMIPS instructions placed in a 16-bit delay slot of a branch or a jump.

Installing multiple targets in one directory. Due to changes in the installer, Sourcery CodeBench Lite for MIPS GNU/Linux can no longer be installed into a directory already containing an existing Sourcery CodeBench installation for a different target. The installer detects this situation and asks you to select a different directory.

Install to empty directory failure fixed. A bug that prevented installation of Sourcery CodeBench Lite into an existing empty directory has been fixed.

Misleading GDB warning fixed. A bug has been fixed that formerly caused GDB to display a warning about being unable to load symbols for ELF libraries that are contained in the Linux kernel itself.

Updated system requirements. The host operating system requirements for Sourcery CodeBench Lite have been updated. The minimum versions of GNU/Linux now supported are Red Hat Enterprise Linux 5, SuSE Enterprise Linux 10, Fedora Core 6, Ubuntu 8.04, and Debian 5, or later versions of these distributions running on 32-bit or 64-bit Intel or AMD CPUs.

A.1.5. Changes in Sourcery CodeBench Lite 2012.09-43

GCC version 4.7.2. Sourcery CodeBench Lite for MIPS GNU/Linux is now based on GCC version 4.7.2. For more information about changes from GCC version 4.6 that was included in previous releases, see `http://gcc.gnu.org/gcc-4.7/changes.html`.

Optimization bug fix. A GCC bug has been fixed that caused incorrect code to be generated for `builtin_unreachable` when optimizing.

Linker crash bug fix. A bug has been fixed that caused the linker to crash when using customized linker scripts without a `.eh_frame_hdr` output section description.

Assembler `%hi` and `%lo` operator bug fixes . Several bugs have been fixed in the handling of `%hi` and `%lo` assembly operators applied to expressions involving forward symbol references. The bugs caused the assembler to fail with a `relocation overflow` error message or to produce incorrect code.

Linker script symbols. The linker now supports a new `HIDDEN` keyword to define symbols with object scope. Refer to the linker manual for details.

microMIPS `SWXC1` instruction encoding bug fix . A bug in the encoding of the microMIPS `SWXC1` instruction has been fixed. The bug caused `LWXC1` to be produced by the assembler and listed by the disassembler instead.

Binutils version 2.23. Sourcery CodeBench Lite for MIPS GNU/Linux is now based on binutils version 2.23.

Linker assertion failure bug fix. A linker bug has been fixed that caused assertion failures when processing code that makes use of protected symbols.

EVA instruction assembler bug fix. An assembler bug in branch delay slot optimization involving EVA instructions has been fixed.

Assembler internal error bug fix. A bug has been fixed that caused the assembler to sometimes fail with a `fixup not contained within frag` error message. The bug was triggered by the branch delay slot optimization, not normally used for compiler-generated code.

Linker `_gp` special symbol ABI conformance bug fix . A bug has been fixed in the linker that caused link errors or run-time malfunction of code that referred to the special `_gp` symbol. The semantics of the symbol has been defined in the MIPS processor-specific ABI supplement and the linker did not follow that definition.

EGLIBC version 2.16. Sourcery CodeBench Lite for MIPS GNU/Linux now includes EGLIBC version 2.16 library which is based on GNU C Library version 2.16. For more information about changes, see `http://www.eglibc.org/news#eglibc_2_16`.

Linux kernel headers update. Linux kernel header files have been updated to version 3.5.4.

GDB update. The included version of GDB has been updated to 7.4.50.20120716. This update adds numerous bug fixes and features. Refer to `http://www.gnu.org/software/gdb/news` for more information.

Non-stop support. The Sourcery CodeBench Debug Sprite now supports non-stop debugging.

A.1.6. Changes in Sourcery CodeBench Lite 2012.03-63

`__attribute__((nomips16))` code generation bug fix . A bug in GCC that caused incorrect code to be generated for functions with the `__attribute__((nomips16))` attribute while compiling with `-mips16` has been fixed.

C++ bug fix. A bug that caused unpredictable program behavior in C++ programs has been fixed.

Invalid microMIPS relocation fixed. A bug in branch code generation for microMIPS, reported by the linker as `relocation truncated to fit: R_MICROMIPS_PC16_S1`, has been fixed.

Improved size optimization for microMIPS. The compiler now aligns microMIPS functions more efficiently when the `-Os` option is used.

Compiler crash fixed. A GCC bug that occasionally caused an internal compiler error during register allocation has been fixed.

Register allocation bug fix. A bug in the register allocator that caused incorrect code generation has been fixed.

Linker `--gc-sections` bug fix. A linker bug that incorrectly caused undefined references to be diagnosed when the `--gc-sections` option is used has been fixed.

EVA support. GAS now supports MIPS EVA instructions. You can specify the `-meva` option to GCC and GAS to indicate that EVA instructions are being used.

RPC library functions. GLIBC has been changed so that programs may again be built to use its RPC library functions. The ability to build programs using these functions had previously been disabled.

A.1.7. Changes in Sourcery CodeBench Lite 2012.03-40

Nondeterministic code generation bug fix. A GCC bug has been fixed that caused nondeterministic code generation for some input files when optimizing.

GCC version 4.6. Sourcery CodeBench Lite for MIPS GNU/Linux is now based on GCC version 4.6. For more information about changes from GCC version 4.5 that was included in previous releases, see `http://gcc.gnu.org/gcc-4.6/changes.html`.

Fix for internal compiler error. A GCC bug has been fixed that caused an internal compiler error when using pointer casts in C++0x `constexpr` initialization expressions.

Compact C++ exception handling tables. GCC now defaults to emitting a compact encoding of C++ exception handling tables, rather than using a DWARF-based scheme. For more information, refer to the documentation of the `-mcompact-eh` and `-mno-compact-eh` options in the GCC manual.

Fix for bit-field optimization bug. A compiler bug that caused incorrect code to be generated for programs using bit-fields has been fixed.

Incorrect accesses to volatile memory. The compiler no longer generates SWP, LWP, SWM or LWM instructions to access objects declared with the `volatile` type qualifier, as these instructions can cause multiple memory accesses of unspecified ordering.

GCC version 4.6.3. Sourcery CodeBench Lite for MIPS GNU/Linux is now based on GCC version 4.6.3. For more information about issues that have been fixed since version 4.6.1, see `http://gcc.gnu.org/gcc-4.6/changes.html`.

GCC stack usage improvement. GCC now generates better code for stack allocation in some cases when compiling with `-fno-strict-aliasing`.

microMIPS PLT linker bug fix. A bug has been fixed that sometimes caused the linker to emit MIPS32 code in the Procedure Linkage Table (PLT) in programs linked from pure microMIPS objects.

Linker `--gc-sections` option bug fix. A bug has been fixed that caused the linker to incorrectly remove the `.debug_types` section when using the `--gc-sections` option.

Fix for assembler internal error. A bug has been fixed that caused the assembler to abort with an assertion failure in `emit_inc_line_addr`.

Installer failure during upgrade. Some recent releases were affected by an installer bug on Windows hosts that caused installing a newer version of Sourcery CodeBench Lite into the same directory to fail with the error `Sourcery CodeBench Lite for MIPS GNU/Linux upgrade failed`. This bug has now been fixed, but the affected releases cannot be upgraded. As a workaround, uninstall the older release before installing the new version.

`zic` bug fix. A bug has been fixed in GLIBC's `zic` command (included in the target sysroot) that caused `too many transitions` errors on some valid input files.

EGLIBC version 2.15. Sourcery CodeBench Lite for MIPS GNU/Linux now includes EGLIBC version 2.15 library which is based on GNU C Library version 2.15. For more information about changes, see `http://www.eglibc.org/news#eglibc_2_15`.

Linux kernel headers update. Linux kernel header files have been updated to version 3.2.10.

C++ debugging bug fix. A GDB bug has been fixed that caused GDB to fail to find `enum` constants in base classes when debugging C++ code.

Fix for crash in GDB. A memory corruption bug in GDB has been fixed that under very rare circumstances made it crash or exhibit other unpredictable behavior. On GNU/Linux hosts, this bug caused crashes with an error message similar to: `*** glibc detected *** mips-linux-gnu-gdb: free(): invalid next size (normal): 0x09466198 ***` followed by a backtrace.

Fix debugger remote target interruption. A bug in GDB's handling of requests to interrupt execution on a remote target has been fixed that caused it to stop the target but not emit a stopped MI record.

GDB internal error fix. A bug has been fixed that caused GDB to produce messages of the form: `warning: (Internal error: pc 0x1000a0 in read in psymtab, but not in symtab.)` when taking the addresses of symbols from objects added with the `add-symbol-file` command.

A.1.8. Changes in Older Releases

For information about changes in older releases of Sourcery CodeBench Lite for MIPS GNU/Linux, please refer to the Getting Started guide packaged with those releases.

Appendix B
Sourcery CodeBench Lite Licenses

Sourcery CodeBench Lite contains software provided under a variety of licenses. Some components are “free” or “open source” software, while other components are proprietary. This appendix explains what licenses apply to your use of Sourcery CodeBench Lite. You should read this appendix to understand your legal rights and obligations as a user of Sourcery CodeBench Lite.

The Mentor Graphics License is available in Section B.1, “Sourcery CodeBench Lite License Agreement”.

B.1. Sourcery CodeBench Lite License Agreement

Sourcery CodeBench Lite for MIPS GNU/Linux is licensed under the Mentor Graphics **Embedded Software and Hardware License Agreement.** If you have a separate signed or shrinkwrap agreement (as applicable) with Mentor Graphics related to your use of Sourcery CodeBench Lite, your order is subject to the terms of that agreement. If you do not, the following terms apply, unless otherwise specifically agreed to in writing by an authorized representative of Mentor Graphics. The terms of this Getting Started Guide supplement, but do not replace or amend, the terms of your separate agreement with Mentor Graphics. Accordingly, to the extent the following terms and conditions conflict with such separate agreement, the terms and conditions of the separate agreement shall control.

The latest version of the License Agreement is available on-line at http://www.mentor.com/terms_conditions/embedded_software_license.

B.1.1. Embedded Software and Hardware License Agreement

IMPORTANT INFORMATION

USE OF ALL PRODUCTS IS SUBJECT TO LICENSE RESTRICTIONS. CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE PRODUCTS. USE OF PRODUCTS INDICATES CUSTOMER'S COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. ANY ADDITIONAL OR DIFFERENT PURCHASE ORDER TERMS AND CONDITIONS SHALL NOT APPLY.

EMBEDDED SOFTWARE AND HARDWARE LICENSE AGREEMENT ("Agreement")

This is a legal agreement concerning the use of Products (as defined in Section 1) between the company acquiring the Products ("Customer"), and the Mentor Graphics entity that issued the corresponding quotation or, if no quotation was issued, the applicable local Mentor Graphics entity ("Mentor Graphics"). Except for license agreements related to the subject matter of this license agreement which are physically signed by Customer and an authorized representative of Mentor Graphics, this Agreement and the applicable quotation contain the parties' entire understanding relating to the subject matter and supersede all prior or contemporaneous agreements. If Customer does not agree to these terms and conditions, promptly return or, in the case of Products received electronically, certify destruction of Products and all accompanying items within five days after receipt of such Products and receive a full refund of any license fee paid.

1. **Definitions.**

    1.1. "Customer's Product" means Customer's end-user product identified by a unique SKU (including any Related SKUs) in an applicable Addenda that is developed, manufactured, branded and shipped solely by Customer or an authorized manufacturer or subcontractor on behalf of Customer to end-users or consumers;

1.2. "Developer" means a unique user, as identified by a unique user identification number, with access to Embedded Software at an authorized Development Location. A unique user is an individual who works directly with the embedded software in source code form, or creates, modifies or compiles software that ultimately links to the Embedded Software in Object Code form and is embedded into Customer's Product at the point of manufacture;

1.3. "Development Location" means the location where Products may be used as authorized in the applicable Addenda;

1.4. "Development Tools" means the software that may be used by Customer for building, editing, compiling, debugging or prototyping Customer's Product;

1.5. "Embedded Software" means Software that is embeddable;

1.6. "End-User" means Customer's customer;

1.7. "Executable Code" means a compiled program translated into a machine-readable format that can be loaded into memory and run by a certain processor;

1.8. "Hardware" means a physically tangible electro-mechanical system or sub-system and associated documentation;

1.9. "Linkable Object Code" or "Object Code" means linkable code resulting from the translation, processing, or compiling of Source Code by a computer into machine-readable format;

1.10. "Mentor Embedded Linux" or "MEL" means Mentor Graphics' tools, source code, and recipes for building Linux systems;

1.11. "Open Source Software" or "OSS" means software subject to an open source license which requires as a condition for redistribution of such software, including modifications thereto, that the: (i) redistribution be in source code form or be made available in source code form; (ii) redistributed software be licensed to allow the making of derivative works; or (iii) redistribution be at no charge;

1.12. "Processor" means the specific microprocessor to be used with Software and implemented in Customer's Product;

1.13. "Products" means Software, Term-Licensed Products and/or Hardware;

1.14. "Proprietary Components" means the components of the Products that are owned and/or licensed by Mentor Graphics and are not subject to an Open Source Software license, as more fully set forth herein;

1.15. "Redistributable Components" means those components that are intended to be incorporated or linked into Customer's Linkable Object Code developed with the Software, as more fully set forth herein;

1.16. "Related SKU" means two or more Customer Products identified by logically-related SKUs, where there is no difference or change in the electrical hardware or software content between such Customer Products;

1.17. "Software" means software programs, Embedded Software and/or Development Tools, including any updates, modifications, revisions, copies, documentation and design data that are licensed under this Agreement;

1.18. "Source Code" means software in a form in which the program logic is readily understandable by a human being;

1.19. "Sourcery CodeBench Software" means Mentor Graphics' Development Tool for C/C++ embedded application development;

1.20. "Sourcery VSIPL++" is Software providing C++ classes and functions for writing embedded signal processing applications designed to run on one or more processors;

1.21. "Stock Keeping Unit" or "SKU" is a unique number or code used to identify each distinct product, item or service available for purchase;

1.22. "Subsidiary" means any corporation more than 50% owned by Customer, excluding Mentor Graphics competitors. Customer agrees to fulfill the obligations of such Subsidiary in the event of default. To the extent Mentor Graphics authorizes any Subsidiary's use of Products under this Agreement, Customer agrees to ensure such Subsidiary's compliance with the terms of this Agreement and will be liable for any breach by a Subsidiary; and

1.23. "Term-Licensed Products" means Products licensed to Customer for a limited time period ("Term").

2. **Orders, Fees and Payment.**

2.1. To the extent Customer (or if agreed by Mentor Graphics, Customer's appointed third party buying agent) places and Mentor Graphics accepts purchase orders pursuant to this Agreement ("Order(s)"), each Order will constitute a contract between Customer and Mentor Graphics, which shall be governed solely and exclusively by the terms and conditions of this Agreement and any applicable Addenda, whether or not these documents are referenced on the Order. Any additional or conflicting terms and conditions appearing on an Order will not be effective unless agreed in writing by an authorized representative of Customer and Mentor Graphics.

2.2. Amounts invoiced will be paid, in the currency specified on the applicable invoice, within 30 days from the date of such invoice. All invoices will be sent electronically to Customer on the date stated on the invoice unless otherwise specified in an Addendum. Any past due invoices will be subject to the imposition of interest charges in the amount of one and one-half percent per month or the applicable legal rate currently in effect, whichever is lower. Prices do not include freight, insurance, customs duties, taxes or other similar charges, which Mentor Graphics will state separately in the applicable invoice(s). Unless timely provided with a valid certificate of exemption or other evidence that items are not taxable, Mentor Graphics will invoice Customer for all applicable taxes including, but not limited to, VAT, GST, sales tax, consumption tax and service tax. Customer will make all payments free and clear of, and without reduction for, any withholding or other taxes; any such taxes imposed on payments by Customer hereunder will be Customer's sole responsibility. If Customer appoints a third party to place purchase orders and/or make payments on Customer's behalf, Customer shall be liable for payment under Orders placed by such third party in the event of default.

2.3. All Products are delivered FCA factory (Incoterms 2010), freight prepaid and invoiced to Customer, except Software delivered electronically, which shall be deemed delivered when made available to Customer for download. Mentor Graphics' delivery of Software by electronic means is subject to Customer's provision of both a primary and an alternate e-mail address.

3. **Grant of License.**

3.1. The Products installed, downloaded, or otherwise acquired by Customer under this Agreement constitute or contain copyrighted, trade secret, proprietary and confidential information of Mentor Graphics or its licensors, who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Mentor Graphics grants to Customer, subject to payment of applicable license fees, a nontransferable, nonexclusive license to use Software as described in the applicable Addenda. The limited licenses granted under the applicable Addenda shall continue until the expiration date of Term-Licensed Products or termination in accordance with Section 12 below, whichever occurs first. Mentor Graphics does NOT grant Customer any right to (a) sublicense or (b) use Software beyond the scope of this Section without first signing a separate agreement or Addenda with Mentor Graphics for such purpose.

3.2. License Type. The license type shall be identified in the applicable Addenda.

3.2.1. Development License: During the Term, if any, Customer may modify, compile, assemble and convert the applicable Embedded Software Source Code into Linkable Object Code and/or Executable Code form by the number of Developers specified, for the Processor(s), Customer's Product(s) and at the Development Location(s) identified in the applicable Addenda.

3.2.2. End-User Product License: During the Term, if any, and unless otherwise specified in the applicable Addenda, Customer may incorporate or embed an Executable Code version of the Embedded Software into the specified number of copies of Customer's Product(s), using the Processor Unit(s), and at the Development Location(s) identified in the applicable Addenda. Customer may manufacture, brand and distribute such Customer's Product(s) worldwide to its End-Users.

3.2.3. Internal Tool License: During the Term, if any, Customer may use the Development Tools solely: (a) for internal business purposes and (b) on the specified number of computer work stations and sites. Development Tools are licensed on a per-seat or floating basis, as specified in the applicable Addenda, and shall not be distributed to others or delivered in Customer's Product(s) unless specifically authorized in an applicable Addenda.

3.2.4. Sourcery CodeBench Professional Edition License: During the Term specified in the applicable Addenda, Customer may (a) install and use the Proprietary Components of the Software (i) if the license is a node-locked license, by a single user who uses the Software on up to two machines provided that only one copy of the Software is in use at any one time, or (ii) if the license is a floating license, by the authorized number of concurrent users on one or more machines provided that only the authorized number of copies of the Software are in use at any one time, and (b) distribute the Redistributable Components of the Software in Executable Code form only and only as part of Customer's Object Code developed with the Software that provides substantially different functionality than the Redistributable Component(s) alone.

3.2.5. Sourcery CodeBench Standard Edition License: During the Term specified in the applicable Addenda, Customer may (a) install and use the Proprietary Components of the Software by a single user who uses the Software on up to two machines provided that only one copy of the Software is in use at any one time, and (b) distribute the Redistributable Component(s) of the Software in Executable Code form only and only as part of Customer's Object Code developed with the Software that provides substantially different functionality than the Redistributable Component(s) alone.

3.2.6. Sourcery CodeBench Personal Edition License: During the Term specified in the applicable Addenda, Customer may (a) install and use the Proprietary Components

of the Software by a single user who uses the Software on one machine, and (b) distribute the Redistributable Component(s) of the Software in Executable Code form only and only as part of Customer Object Code developed with the Software that provides substantially different functionality than the Redistributable Component(s) alone.

3.2.7. Sourcery CodeBench Academic Edition License: During the Term specified in the applicable Addenda, Customer may (a) install and use the Proprietary Components of the Software for non-commercial, academic purposes only by a single user who uses the Software on one machine, and (b) distribute the Redistributable Component(s) of the Software in Executable Code form only and only as part of Customer Object Code developed with the Software that provides substantially different functionality than the Redistributable Component(s) alone.

3.3. Mentor Graphics may from time to time, in its sole discretion, lend Products to Customer. For each loan, Mentor Graphics will identify in writing the quantity and description of Software loaned, the authorized location and the Term of the loan. Mentor Graphics will grant to Customer a temporary license to use the loaned Software solely for Customer's internal evaluation in a non-production environment. Customer shall return to Mentor Graphics or delete and destroy loaned Software on or before the expiration of the loan Term. Customer will sign a certification of such deletion or destruction if requested by Mentor Graphics.

4. **Beta Code.**

4.1. Portions or all of certain Products may contain code for experimental testing and evaluation ("Beta Code"), which may not be used without Mentor Graphics' explicit authorization. Upon Mentor Graphics' authorization, Mentor Graphics grants to Customer a temporary, nontransferable, nonexclusive license for experimental use to test and evaluate the Beta Code without charge for a limited period of time specified by Mentor Graphics. This grant and Customer's use of the Beta Code shall not be construed as marketing or offering to sell a license to the Beta Code, which Mentor Graphics may choose not to release commercially in any form.

4.2. If Mentor Graphics authorizes Customer to use the Beta Code, Customer agrees to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. Customer will contact Mentor Graphics periodically during Customer's use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of Customer's evaluation and testing, Customer will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements.

4.3. Customer agrees to maintain Beta Code in confidence and shall restrict access to the Beta Code, including the methods and concepts utilized therein, solely to those employees and Customer location(s) authorized by Mentor Graphics to perform beta testing. Customer agrees that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceived or made during or subsequent to this Agreement, including those based partly or wholly on Customer's feedback, will be the exclusive property of Mentor Graphics. Mentor Graphics will have exclusive rights, title and interest in all such property. The provisions of this Subsection 4.3 shall survive termination of this Agreement.

5. **Restrictions on Use.**

5.1. Customer may copy Software only as reasonably necessary to support the authorized use, including archival and backup purposes. Each copy must include all notices and legends

embedded in Software and affixed to its medium and container as received from Mentor Graphics. All copies shall remain the property of Mentor Graphics or its licensors. Except where embedded in Executable Code form in Customer's Product, Customer shall maintain a record of the number and location of all copies of Software, including copies merged with other software and products, and shall make those records available to Mentor Graphics upon request. Customer shall not make Products available in any form to any person other than Customer's employees, authorized manufacturers or authorized contractors, excluding Mentor Graphics competitors, whose job performance requires access and who are under obligations of confidentiality. Customer shall take appropriate action to protect the confidentiality of Products and ensure that any person permitted access does not disclose or use Products except as permitted by this Agreement. Customer shall give Mentor Graphics immediate written notice of any unauthorized disclosure or use of the Products as soon as Customer learns or becomes aware of such unauthorized disclosure or use.

5.2. Customer acknowledges that the Products provided hereunder may contain Source Code which is proprietary and its confidentiality is of the highest importance and value to Mentor Graphics. Customer acknowledges that Mentor Graphics may be seriously harmed if such Source Code is disclosed in violation of this Agreement. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, Customer shall not reverse-assemble, reverse-compile, reverse-engineer or in any way derive any Source Code from Products that are not provided in Source Code form. Except as embedded in Executable Code in Customer's Product and distributed in the ordinary course of business, in no event shall Customer provide Products to Mentor Graphics competitors. Log files, data files, rule files and script files generated by or for the Software (collectively "Files") constitute and/or include confidential information of Mentor Graphics. Customer may share Files with third parties, excluding Mentor Graphics competitors, provided that the confidentiality of such Files is protected by written agreement at least as well as Customer protects other information of a similar nature or importance, but in any case with at least reasonable care. Under no circumstances shall Customer use Products or allow their use for the purpose of developing, enhancing or marketing any product that is in any way competitive with Products, or disclose to any third party the results of, or information pertaining to, any benchmark.

5.3. Customer may not assign this Agreement or the rights and duties under it, or relocate, sublicense or otherwise transfer the Products, whether by operation of law or otherwise ("Attempted Transfer"), without Mentor Graphics' prior written consent, which shall not be unreasonably withheld, and payment of Mentor Graphics' then-current applicable relocation and/or transfer fees. Any Attempted Transfer without Mentor Graphics' prior written consent shall be a material breach of this Agreement and may, at Mentor Graphics' option, result in the immediate termination of the Agreement and/or the licenses granted under this Agreement. The terms of this Agreement, including without limitation the licensing and assignment provisions, shall be binding upon Customer's permitted successors in interest and assigns.

5.4. Notwithstanding any provision in an OSS license agreement applicable to a component of the Sourcery CodeBench Software that permits the redistribution of such component to a third party in Source Code or binary form, Customer may not use any Mentor Graphics trademark, whether registered or unregistered, in connection with such distribution, and may not recompile the Open Source Software components with the --with-pkgversion or --with-bugurl configuration options that embed Mentor Graphics' trademarks in the resulting binary.

5.5. The provisions of this Section 5 shall survive the termination of this Agreement.

6. **Support Services.**

   6.1. Except as described in Sections 6.2, 6.3 and 6.4 below, and unless otherwise specified in any applicable Addenda to this Agreement, to the extent Customer purchases support services, Mentor Graphics will provide Customer updates and technical support for the number of Developers at the Development Location(s) for which support is purchased in accordance with Mentor Graphics' then-current End-User Software Support Terms located at http://supportnet.mentor.com/about/legal/.

   6.2. To the extent Customer purchases support services for Sourcery CodeBench Software, support will be provided solely in accordance with the provisions of this Section 6.2. Mentor Graphics shall provide updates and technical support to Customer as described herein only on the condition that Customer uses the Executable Code form of the Sourcery CodeBench Software for internal use only and/or distributes the Redistributable Components in Executable Code form only (except as provided in a separate redistribution agreement with Mentor Graphics or as required by the applicable Open Source license). Any other distribution by Customer of the Sourcery CodeBench Software (or any component thereof) in any form, including distribution permitted by the applicable Open Source license, shall automatically terminate any remaining support term. Subject to the foregoing and the payment of support fees, Mentor Graphics will provide Customer updates and technical support for the number of Developers at the Development Location(s) for which support is purchased in accordance with Mentor Graphics' then-current Sourcery CodeBench Software Support Terms located at http://www.mentor.com/codebench-support-legal.

   6.3. To the extent Customer purchases support services for Sourcery VSIPL++, Mentor Graphics will provide Customer updates and technical support for the number of Developers at the Development Location(s) for which support is purchased solely in accordance with Mentor Graphics' then-current Sourcery VSIPL++ Support Terms located at. http://www.mentor.com/vsipl-support-legal.

   6.4. To the extent Customer purchases support services for Mentor Embedded Linux, Mentor Graphics will provide Customer updates and technical support for the number of Developers at the Development Location(s) for which support is purchased solely in accordance with Mentor Graphics' then-current Mentor Embedded Linux Support Terms located at http://www.mentor.com/mel-support-legal.

7. **Third Party and Open Source Software.** Products may contain Open Source Software or code distributed under a proprietary third party license agreement. Please see applicable Products documentation, including but not limited to license notice files, header files or source code for further details. Please see Section B.2.2, “Components” for additional rights and obligations governing your use and distribution of Open Source Software. Customer agrees that it shall not subject any Product provided by Mentor Graphics under this Agreement to any Open Source Software license that does not otherwise apply to such Product. In the event of conflict between the terms of this Agreement, any Addenda and an applicable OSS or proprietary third party agreement, the OSS or proprietary third party agreement will control solely with respect to the OSS or proprietary third party software component. The provisions of this Section 7 shall survive the termination of this Agreement.

8. **Limited Warranty.**

   8.1. Mentor Graphics warrants that during the warranty period its standard, generally supported Products, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual and/or specification. Mentor Graphics does not warrant that Products will meet Customer's requirements or that operation of Products will be uninterrupted or error free. The warranty period is 90 days starting on the 15th day after

delivery or upon installation, whichever first occurs. Customer must notify Mentor Graphics in writing of any nonconformity within the warranty period. For the avoidance of doubt, this warranty applies only to the initial shipment of Products under an Order and does not renew or reset, for example, with the delivery of (a) Software updates or (b) authorization codes. This warranty shall not be valid if Products have been subject to misuse, unauthorized modification or improper installation. MENTOR GRAPHICS' ENTIRE LIABILITY AND CUSTOMER'S EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS' OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF THE PRODUCTS TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF THE PRODUCTS THAT DO NOT MEET THIS LIMITED WARRANTY, PROVIDED CUSTOMER HAS OTHERWISE COMPLIED WITH THIS AGREEMENT. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; OR (B) PRODUCTS PROVIDED AT NO CHARGE, WHICH ARE PROVIDED "AS IS" UNLESS OTHERWISE AGREED IN WRITING.

8.2. THE WARRANTIES SET FORTH IN THIS SECTION 8 ARE EXCLUSIVE TO CUSTOMER AND DO NOT APPLY TO ANY END-USER. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, WITH RESPECT TO PRODUCTS OR OTHER MATERIAL PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.

9. **LIMITATION OF LIABILITY.** EXCEPT WHERE THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE LAW, AND EXCEPT FOR EITHER PARTY'S BREACH OF ITS CONFIDENTIALITY OBLIGATIONS, CUSTOMER'S BREACH OF LICENSING TERMS OR CUSTOMER'S OBLIGATIONS UNDER SECTION 10, IN NO EVENT SHALL: (A) EITHER PARTY OR ITS RESPECTIVE LICENSORS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL THEORY, EVEN IF SUCH PARTY OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES; AND (B) EITHER PARTY OR ITS RESPECTIVE LICENSORS' LIABILITY UNDER THIS AGREEMENT, INCLUDING, FOR THE AVOIDANCE OF DOUBT, LIABILITY FOR ATTORNEYS' FEES OR COSTS, EXCEED THE GREATER OF THE FEES PAID OR OWING TO MENTOR GRAPHICS FOR THE PRODUCT OR SERVICE GIVING RISE TO THE CLAIM OR $500,000 (FIVE HUNDRED THOUSAND U.S. DOLLARS). NOTWITHSTANDING THE FOREGOING, IN THE CASE WHERE NO AMOUNT WAS PAID, MENTOR GRAPHICS AND ITS LICENSORS SHALL HAVE NO LIABILITY FOR ANY DAMAGES WHATSOEVER. THE PROVISIONS OF THIS SECTION 9 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

10. **Hazardous Applications.**

10.1. Customer agrees that Mentor Graphics has no control over Customer's testing or the specific applications and use that Customer will make of Products. Mentor Graphics Products are not specifically designed for use in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, life support systems, medical devices or other applications in which the failure of Mentor Graphics Products could lead to death, personal injury, or severe physical or environmental damage ("Hazardous Applications").

10.2. CUSTOMER ACKNOWLEDGES IT IS SOLELY RESPONSIBLE FOR TESTING PRODUCTS USED IN HAZARDOUS APPLICATIONS AND SHALL BE SOLELY

LIABLE FOR ANY DAMAGES RESULTING FROM SUCH USE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS SHALL BE LIABLE FOR ANY DAMAGES RESULTING FROM OR IN CONNECTION WITH THE USE OF PRODUCTS IN ANY HAZARDOUS APPLICATIONS.

10.3. CUSTOMER AGREES TO INDEMNIFY AND HOLD HARMLESS MENTOR GRAPHICS AND ITS LICENSORS FROM ANY CLAIMS, LOSS, COST, DAMAGE, EXPENSE OR LIABILITY, INCLUDING REASONABLE ATTORNEYS' FEES, ARISING OUT OF OR IN CONNECTION WITH THE USE OF PRODUCTS AS DESCRIBED IN SECTION 10.1.

10.4. THE PROVISIONS OF THIS SECTION 10 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

11. **Infringement.**

11.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against Customer in the United States, Canada, Japan, or member state of the European Union which alleges that any standard, generally supported Product acquired by Customer hereunder infringes a patent or copyright or misappropriates a trade secret in such jurisdiction. Mentor Graphics will pay any costs and damages finally awarded against Customer that are attributable to the action. Customer understands and agrees that as conditions to Mentor Graphics' obligations under this section Customer must: (a) notify Mentor Graphics promptly in writing of the action; (b) provide Mentor Graphics all reasonable information and assistance to settle or defend the action; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the action.

11.2. If a claim is made under Subsection 11.1 Mentor Graphics may, at its option and expense, and in addition to its obligations under Section 11.1, either (a) replace or modify the Product so that it becomes noninfringing; or (b) procure for Customer the right to continue using the Product. If Mentor Graphics determines that neither of those alternatives is financially practical or otherwise reasonably available, Mentor Graphics may require the return of the Product and refund to Customer any purchase price or license fee(s) paid.

11.3. Mentor Graphics has no liability to Customer if the claim is based upon: (a) the combination of the Product with any product not furnished by Mentor Graphics, where the Product itself is not infringing; (b) the modification of the Product other than by Mentor Graphics or as directed by Mentor Graphics, where the unmodified Product would not infringe; (c) the use of the infringing Product when Mentor Graphics has provided Customer with a current unaltered release of a non-infringing Product of substantially similar functionality in accordance with Subsection 11.2(a); (d) the use of the Product as part of an infringing process; (e) a product that Customer makes, uses, or sells, where the Product itself is not infringing; (f) any Product provided at no charge; (g) any software provided by Mentor Graphics' licensors who do not provide such indemnification to Mentor Graphics' customers; (h) Open Source Software, except to the extent that the infringement is directly caused by Mentor Graphics' modifications to such Open Source Software; or (i) infringement by Customer that is deemed willful. In the case of (i), Customer shall reimburse Mentor Graphics for its reasonable attorneys' fees and other costs related to the action.

11.4. THIS SECTION 11 IS SUBJECT TO SECTION 9 ABOVE AND STATES: (A) THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS AND (B) CUSTOMER'S SOLE AND EXCLUSIVE REMEDY, WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT OR TRADE SECRET MISAPPROPRIATION BY ANY PRODUCT PROVIDED UNDER THIS AGREEMENT.

12. **Termination and Effect of Termination.** If a Software license was provided for limited term use, such license will automatically terminate at the end of the authorized Term.

    12.1. **Termination for Breach.** This Agreement shall remain in effect until terminated in accordance with its terms. Mentor Graphics may terminate this Agreement and/or any licenses granted under this Agreement, and Customer will immediately discontinue use and distribution of Products, if Customer (a) commits any material breach of any provision of this Agreement and fails to cure such breach upon 30-days prior written notice; or (b) becomes insolvent, files a bankruptcy petition, institutes proceedings for liquidation or winding up or enters into an agreement to assign its assets for the benefit of creditors. Termination of this Agreement or any license granted hereunder will not affect Customer's obligation to pay for Products shipped or licenses granted prior to the termination, which amounts shall be payable immediately upon the date of termination. For the avoidance of doubt, nothing in this Section 12 shall be construed to prevent Mentor Graphics from seeking immediate injunctive relief in the event of any threatened or actual breach of Customer's obligations hereunder.

    12.2. **Effect of Termination.** Upon termination of this Agreement, the rights and obligations of the parties shall cease except as expressly set forth in this Agreement. Upon termination or expiration of the Term, Customer will discontinue use and/or distribution of Products, and shall return Hardware and either return to Mentor Graphics or destroy Software in Customer's possession, including all copies and documentation, and certify in writing to Mentor Graphics within ten business days of the termination date that Customer no longer possesses any of the affected Products or copies of Software in any form, except to the extent an Open Source Software license conflicts with this Section 12.2 and permits Customer's continued use of any Open Source Software portion or component of a Product. Upon termination for Customer's breach, an End-User may continue its use and/or distribution of Customer's Product so long as: (a) the End-User was licensed according to the terms of this Agreement, if applicable to such End-User, and (b) such End-User is not in breach of its agreement, if applicable, nor a party to Customer's breach.

13. **Export.** The Products provided hereunder are subject to regulation by local laws and United States government agencies, which prohibit export or diversion of certain products, information about the products, and direct or indirect products thereof, to certain countries and certain persons. Customer agrees that it will not export Products in any manner without first obtaining all necessary approval from appropriate local and United States government agencies. Customer acknowledges that the regulation of product export is in continuous modification by local governments and/or the United States Congress and administrative agencies. Customer agrees to complete all documents and to meet all requirements arising out of such modifications.

14. **U.S. Government License Rights.** Software was developed entirely at private expense. All Software is commercial computer software within the meaning of the applicable acquisition regulations. Accordingly, pursuant to US FAR 48 CFR 12.212 and DFAR 48 CFR 227.7202, use, duplication and disclosure of the Software by or for the U.S. Government or a U.S. Government subcontractor is subject solely to the terms and conditions set forth in this Agreement, except for provisions which are contrary to applicable mandatory federal laws.

15. **Third Party Beneficiary.** For any Products licensed under this Agreement and provided by Customer to End-Users, Mentor Graphics or the applicable licensor is a third party beneficiary of the agreement between Customer and End-User. Mentor Graphics Corporation, Mentor Graphics (Ireland) Limited, and other licensors may be third party beneficiaries of this Agreement with the right to enforce the obligations set forth herein.

16. **Review of License Usage.** Customer will monitor the access to and use of Software. With prior written notice, during Customer's normal business hours, and no more frequently than

once per calendar year, Mentor Graphics may engage an internationally recognized accounting firm to review Customer's software monitoring system, records, accounts and sublicensing documents deemed relevant by the internationally recognized accounting firm to confirm Customer's compliance with the terms of this Agreement or U.S. or other local export laws. Such review may include FlexNet (or successor product) report log files that Customer shall capture and provide at Mentor Graphics' request. Customer shall make records available in electronic format and shall fully cooperate with data gathering to support the license review. Mentor Graphics shall bear the expense of any such review unless a material non-compliance is revealed. Mentor Graphics shall treat as confidential information all Customer information gained as a result of any request or review and shall only use or disclose such information as required by law or to enforce its rights under this Agreement. Such license review shall be at Mentor Graphics' expense unless it reveals a material underpayment of fees of five percent or more, in which case Customer shall reimburse Mentor Graphics for the costs of such license review. Customer shall promptly pay any such fees. If the license review reveals that Customer has made an overpayment, Mentor Graphics has the option to either provide the Customer with a refund or credit the amount overpaid to Customer's next payment. The provisions of this Section 16 shall survive the termination of this Agreement.

17. **Controlling Law, Jurisdiction and Dispute Resolution.** This Agreement shall be governed by and construed under the laws of the State of California, USA, excluding choice of law rules. All disputes arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of the state and federal courts of California, USA. Nothing in this section shall restrict Mentor Graphics' right to bring an action (including for example a motion for injunctive relief) against Customer or its Subsidiary in the jurisdiction where Customer's or its Subsidiary's place of business is located. The United Nations Convention on Contracts for the International Sale of Goods does not apply to this Agreement.

18. **Severability.** If any provision of this Agreement is held by a court of competent jurisdiction to be void, invalid, unenforceable or illegal, such provision shall be severed from this Agreement and the remaining provisions will remain in full force and effect.

19. **Miscellaneous.** This Agreement contains the parties' entire understanding relating to its subject matter and supersedes all prior or contemporaneous agreements, including but not limited to any purchase order terms and conditions. This Agreement may only be modified in writing, signed by an authorized representative of each party. Waiver of terms or excuse of breach must be in writing and shall not constitute subsequent consent, waiver or excuse.

Rev. 120305, Part No. 252061

B.2. Licenses for Sourcery CodeBench Lite Components

The table below lists the major components of Sourcery CodeBench Lite for MIPS GNU/Linux and the license terms which apply to each of these components.

B.2.1. Mentor Graphics Proprietary Components

Components of the Software that are owned and/or licensed by Mentor Graphics and are not subject to a “free software” or “open source” license, such as the GNU Public License. The Mentor Graphics Proprietary Components of the Software include, without limitation, the Sourcery CodeBench Installer, any Sourcery CodeBench Eclipse plug-ins, the CodeSourcery C Library (CSLIBC), and any Sourcery CodeBench Debug Sprite.

B.2.2. Components

Some free or open-source components provide documentation or other files under terms different from those shown below. For definitive information about the license that applies to each component, consult the source package corresponding to this release of Sourcery CodeBench Lite. Sourcery CodeBench Lite may contain free or open-source components not included in the list below; for a definitive list, consult the source package corresponding to this release of Sourcery CodeBench Lite.

Component	**License**
GNU Compiler Collection	GNU General Public License 3.0 http://www.gnu.org/licenses/gpl.html
GNU Binary Utilities	GNU General Public License 3.0 http://www.gnu.org/licenses/gpl.html
GNU Debugger	GNU General Public License 3.0 http://www.gnu.org/licenses/gpl.html
Sourcery CodeBench Debug Sprite	Mentor Graphics License
QEMU Emulator	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
GNU C Library	GNU Lesser General Public License 2.1 http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html
uClibc C Library	GNU Lesser General Public License 2.1 http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html
Linux Kernel Headers	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
GNU Make	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
GNU Core Utilities	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
GNU/Linux Prelinker	GNU General Public License 2.0 http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

Important

Although some of the licenses that apply to Sourcery CodeBench Lite are “free software” or “open source software” licenses, none of these licenses impose any obligation on you to reveal the source code of applications you build with Sourcery CodeBench Lite. You can develop proprietary applications and libraries with Sourcery CodeBench Lite.

Sourcery CodeBench Lite may include some third party example programs and libraries in the `share/sourceryg++-mips-linux-gnu-examples` subdirectory. These examples are not covered by the Sourcery CodeBench Software License Agreement. To the extent permitted by law, these examples are provided by CodeSourcery as is with no warranty of any kind, including implied warranties of merchantability or fitness for a particular purpose. Your use of each example is governed by the license notice (if any) it contains.

B.2.3. Third-Party Information

B.3. Attribution

This version of Sourcery CodeBench Lite may include code based on work under the following copyright and permission notices:

B.3.1. Android Open Source Project

```
/*
 * Copyright (C) 2008 The Android Open Source Project
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *  * Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *  * Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
 * COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
 * BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
 * OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
 * AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
 * OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */
```