Linux/MIPS HOWTO

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Linux/MIPS HOWTO

Ralf Bächle, ralf@gnu.org

v1.65, 2002−08−23

This FAQ describes the MIPS port of the Linux operating system, common problems and their solutions, availability and more. It also tries to be helpful to other people who might read this FAQ in an attempt to find information that actually should be covered elsewhere.

8. Supported Hardware platforms

9.

10. Technical FAQ

11. Documentation

12. Legal Notices

1. The home of the Linux MIPS port

Linux/MIPS is a port of the widespread UNIX clone Linux to the MIPS architecture. Linux/MIPS runs on a large number of technically very different systems ranging from small embedded systems to large desktop machines and servers from SGI and DEC.

Here you will find links to all the resources you need to work with Linux on MIPS, whether you are starting out getting Linux running on your own platform, or developing an end user application or product based on a Linux/MIPS system.

If you are looking for a commercial Linux product with associated support, take a look at the Commercial Linux page. MIPS Technologies also maintain a list of links here to companies providing the MIPS community with tools and services.

If you have input regarding the contents of these pages, see the Documentation page for contact info. Webserver contact is webmaster@linux−mips.org.

---

2. Mailing lists

2.1 Email lists

There are two Linux/MIPS−oriented mailing lists:

linux−mips@linux−mips.org
: This mailing list currently has the most traffic. It is especially of interest as a good number of active developers are subscribed to this list. Subscription to this list is handled via Ecartis (ecartis@linux−mips.org), just send an email with the words *subscribe linux−mips*. In order to unsubscribe, send *unsubscribe linux−mips*. For more, information see also http://oss.sgi.com/mips/mail.html.

  The archives for this list are located at http://oss.sgi.com/mips/archive/

linux−cvs@linux−mips.org
: This is an announcement only mailing list to which a message for every CVS commit into oss.sgi.com, the central CVS archive of the Linux/MIPS community, is being sent. This allows following the development as it happens. Subscription to this list is handled via Ecartis (ecartis@linux−mips.org); just send an email with the words *subscribe linux−cvs*. In order to unsubscribe, send *unsubscribe linux−cvs* to the same address.

2.2 IRC channel.

There is an IRC channel named #mipslinux for Linux/MIPS which may be found on irc.openprojects.net.

---

3. Kernel

The current version of the Linux kernel can be found on kernel.org and will tend to be somewhat ahead of the MIPS/Linux version (see CVS below) but behind in some MIPS−specific regards. Older and more stable versions of the kernel for MIPS are available for download − see the section on Distributions for locations.

3.1 Anonymous CVS servers.

For those who always want to stay on the bleeding edge, and want to avoid having to download patch files or full tarballs, we also have an anonymous CVS server. Using CVS, you can checkout the Linux/MIPS source tree with the following commands:

```
cvs −d :pserver:cvs@oss.sgi.com:/cvs login
(Only needed the first time you use anonymous CVS, the password is "cvs")
cvs −d :pserver:cvs@oss.sgi.com:/cvs co <repository>
```

where you insert linux, libc, gdb or faq for <repository>.

There is a mailing list for information on what gets committed to this repository.

3.2 Web CVS server.

Via http://oss.sgi.com/mips/cvsweb, you have direct access to the new Linux/MIPS kernel sources, and a few other projects hosted in the same CVS archive. The intuitive interface allows you to follow the development at the click of your mouse.

---

4. Distributions.

A distribution is a complete collection of kernel, user programs, toolchain and libraries necessary to get a system up and running. It may include an install procedure to get the files copied correctly onto the target storage device. See the READMEs on the links below for more information.

4.1 Commercial Embedded

See the section on commercial distributions.

4.2 RedHat 7.1 based

A complete distribution based on RedHat's 7.1, ported by H.J. Lu, can be found at ftp://oss.sgi.com/pub/linux/mips/redhat/7.1/.

4.3 Maciej W. Rozyki

A little−endian only distribution is maintained by Maciej at ftp://ftp.ds2.pg.gda.pl/pub/macro/ or the mirror.

4.4 MIPS Technologies

MIPS maintain a version of the above, including complete installable CD−ROM images, at ftp://ftp.mips.com/pub/linux/mips/installation/redhat7.1/.

4.5 Debian

A Debian distribution for both little and big endian machines can be found at http://www.debian.org/ports/mips/. At the time of writing (January 2002) we are using a 2.4 kernel; kernel code is shared with the ports being done by people from MIPS Technologies, Inc.).

4.6 Algorithmics

Algorithmics' kernels and a link to the MIPS userland can be found from a jump page at http://www.algor.co.uk/algor/info/linux.html. Algorithmics wrote the floating point trap handler and emulator used in this kernel − essential for MIPS CPUs to run floating point operations reliably and correctly.

Algorithmics also maintain a GNU toolchain and provide both free snapshots and a commercially supported version − worth thinking about for commercial Linux developments. You can contact Algorithmics at mailto:ask@algor.co.uk.

---

5. Toolchains

A toolchain is a complete collection of compiler and binutils programs and can be run either as a cross−compiler, or native on the target (if performance allows).

5.1 Algorithmics SDE

Not a complete distribution, just a Linux toolchain. But it's a toolchain built and maintained for MIPS with commercial support available, and free snapshots.

Algorithmics specialize in MIPS support and maintain our own source tree for the toolchain components. They resynchronize infrequently with mainstream GNU releases (which inevitably have bugs for minor architectures such as MIPS) and focus on providing the most reliable, best−performing compiler for the largest range of MIPS CPUs.

This is the same compiler which is at the heart of our SDE−MIPS embedded toolkit, and is fully supported for Linux kernel and application building from http://www.algor.co.uk/algor/info/sde5.html v5.0, out in mid 2002.

5.2 Commercial Embedded

See the section on commercial distributions − these all include appropriate toolchain deliverables.

5.3 H.J. Lu

H.J. Lu distributes a toolchain as part of his Red Hat 7.1 port. It can be found at ftp://oss.sgi.com/pub/linux/mips/redhat/7.1/.

5.4 Maciej W. Rozycki

A stable set of toolchain components provided by Maciej can be downloaded from ftp://ftp.ds2.pg.gda.pl/pub/macro/ or the mirror. This is based on gcc 2.95.3 (patched) and up−to−date binutils.

---

6. Commercial Linux

The following companies provide commercial, supported Linux/MIPS solutions for the embedded market, on a number of different platforms.

- MontaVista
- Red Hat
- Lineo
- LynuxWorks
- TimeSys

---

7. Web resources

7.1 Webservers

The following webservers are relevant for Linux/MIPS developers.

http://oss.sgi.com/mips
: This server covers most of Linux/MIPS. If you need something chances are it's already there.

http://www.mips.com/devTools/devArea/Linux.html
: This sites has MIPS own version of Linux/MIPS based distributions and tools for their processors and evaluation boards.

http://www.debian.org/ports/mips/
: This is Debian's MIPS/Linux site.

http://www.playstation2−linux.com
: This is Sony's Linux/MIPS server for the Playstation 2. Also in Japanese.

http://www.ltc.com/~brad/mips/mips−cross−toolchain/
: Bradley D. LaRonde's HowTo on building a cross toolchain for MIPS/Linux.

http://www.junsun.net/linux/porting−howto/
: Jun Sun's porting guide has some useful information and tips for porting to new platforms.

7.2 Anonymous FTP servers.

The primary anonymous FTP servers for Linux/MIPS are

ftp://ftp.linux−mips.org
: This server should satisfy almost all of your Linux/MIPS related ftp desires. Really.

ftp://ftp.mips.com/pub/linux
: This is the server of MIPS, Inc. itself. Among other things it offers a recent Redhat−based distribution and a support area for MIPS' evaluation boards.

ftp://ftp.ds2.pg.gda.pl/pub/macro/
: Maciej W. Rozycki's FTP server.

---

8. Supported Hardware platforms

Check also the section on Supported CPUs for which processor types are supported, if you are using a platform for which multiple CPU options exist.

See below for the following categories of platforms

- Actively supported development boards
- Actively supported workstations
- Legacy only / unsupported
- Never to be supported

8.1 Actively supported development boards

The following list covers development boards for which there is active support for the port, and which are maintained continuously. Expect these ports to work reliably. Refer also the the section on commercial Linux ports − these companies may provide additional hardware support.

Algorithmics P−6032 and P−6064 (and P−4032, P−5064)

Algorithmics (http://www.algor.co.uk/) make a series of single−board computers for MIPS prototyping, and maintain Linux kernels for all of them:

- http://www.algor.co.uk/algor/info/p6032.htmlP−6032 is a board for CPUs with 32−bit buses (PMC−Sierra RM5231, NEC Vr43x0, NEC Vr5432, IDT 64x74)
- http://www.algor.co.uk/algor/info/p6064.htmlP−6064 is for CPUs with 64−bit buses, notably QED's RM70xx and NEC's Vr5500.
- P−4032 is an older board obsoleted by P−6032; P−5064 is obsoleted by P−6064. Linux kernels for both are probably available, but not up to date.

All the boards have common I/O plus Ethernet and disk interfaces onboard, with spare PCI slots for adding different controllers. They're highly configurable, so will run with either byte order. All are suitable targets for 64−bit kernels, but (so far) all the Linux work we've done has been using 32−bit code.

They're available, supported and documented with PDF manuals available online, like http://www.algor.co.uk/ftp/pub/doc/p6032−user.pdf for the P−6032.

Broadcom BCM91250A

An evaluation board for the SiByteTM BCM1250 dual processor SOC (system on chip) and is implemented in the standard ATX form factor. A high performance board. See http://www.broadcom.com/ for details.

MIPS Malta

The MIPS Technologies Malta board and all its CPU options are supported. See the Developers pages under www.mips.com.

8.2 Actively supported workstations

The following list covers machines for which there is active support for the port, and which are maintained continuously. Expect these ports to work reliably.

Cobalt Qube and Raq

The Cobalt Qube product series are low cost headless server systems based on a IDT R5230. Cobalt has developed its own Linux/MIPS variant to fit the special requirements of the Qube as well as possible. Cobalt kernels are available from Cobalt's ftp site http://www.cobaltnet.com.

DECstation series

The following DECstation models are actively supported:

- 2100, codename PMAX
- 5000/xx (Personal DECstation), codename MAXine
- 5000/1xx, codename 3MIN
- 5000/200, codename 3MAX
- 5000/2x0, codename 3MAX+
- 5900/2x0 (identical to the 3MAX+).

These days, most of the work is done by Harald Koerfgen (hkoerfg@web.de) and others. On the Internet, DECstation−related information can be found at http://decstation.unix−ag.org/.

The DECstation family ranges from the DECstation 2100 with an R2000/R2010 chipset at 12 MHz, to the DECstation 5000/260 with a 60 MHz R4400SC. See the section on Legacy Platforms below for other DEC machines. Note: Other x86 and Alpha−based machines were also sold under the name DECstation.

Silicon Graphics Indy

The Indy is currently the best supported Silicon Graphics machine.

Silicon Graphics Origin 200 and 2000

Ralf Bächle (ralf@gnu.org) and a team of SGI employees are currently working on a port to the Origin 200. While still in it's early stages, it's running in uniprocessor and multiprocessor mode and has drivers for the built−in IOC3 Ethernet and SCSI hostadapters.

Sony Playstation and Playstation 2

The Sony Playstation 2 has a Japanese−only port which can be found at http://www.ps2linux.com.

The older Sony Playstation is based on an R3000 derivative and uses a set of graphics chips developed by Sony themselves. There is no support for this machine.

8.3 Unsupported / Legacy support only

The platforms listed here may once have been supported, but there may not have been active maintenance for them. Expect problems with these platforms, and consult the mailing list for information on them.

Acer PICA

The *Acer PICA* is derived from the *Mips Magnum 4000* design. It has a R4400PC CPU running at 133MHz or optionally 150MHz plus a 512KB (optionally 2MB) second level cache; the Magnum's G364 gfx card was replaced with a S3 968 based one. The system is supported with the exception of the X server.

Baget/MIPS series

The Baget series includes several boxes which have R3000 processors: Baget 23, Baget 63, and Baget 83. Baget 23 and 63 have BT23−201 or BT23−202 motherboards with R3500A (which is basically a R3000A chip) at 25 MHz and R3081E at 50 MHz respectively. The BT23−201 board has VME bus and VIC068, VAC068 chips as system controllers. The BT23−202 board has PCI as internal bus and VME as internal. Support for BT23−201 board has been done by Gleb Raiko (rajko@mech.math.msu.su) and Vladimir Roganov (vroganov@msiu.ru) with a bit of help from Serguei Zimin (zimin@msiu.ru). Support for BT23−202 is under development along with Baget 23B which consists of 3 BT23−201 boards with shared VME bus.

Baget 83 is mentioned here for completeness only. It has only 2MB RAM and it's too small to run Linux. The Baget/MIPS code has been merged with the DECstation port. The source for both is available at http://decstation.unix−ag.org/.

DECstation

These DECstation models are orphaned because nobody is working on them, but support for them should be relatively easy to achieve.

- 3100, identical to the 2100 except the R2000A/R2010A @ 16 MHz
- 5100, codename MIPSMATE, almost identical to the 2100 but with an R3000/R3010 chipset.

The other members of the DECstation family, besides the x86 based ones, should be considered as VAXen with the CPU replaced by a MIPS CPU. There is absolutely no information available about these machines and support for them is unlikely to ever happen unless the VAXLinux port comes back to life. These are:

- 5400, codename MIPSFAIR
- 5500, codename MIPSFAIR2
- 5800, codename ISIS

MIPS Magnum 4000 / Olivetti M700−10

These two machines are almost completely identical. Back during the ACE initiative, Olivetti licensed the Jazz design and marketed the machine with Windows NT as the OS. MIPS Computer Systems, Inc. bought

the Jazz design and marketed it as the MIPS Magnum 4000 series of machines. Magnum 4000 systems were marketed with Windows NT and RISC/os as the operating systems.

The firmware on the machine depended on the operating system which was installed. Linux/MIPS supports only the little endian firmware on these two types of machines. Since the M700−10 was only marketed as an NT machine, all M700−10 machines have this firmware installed. The MIPS Magnum case is somewhat more complex. If your machine has been configured big endian for RISC/os, then you need to reload the little endian firmware. This firmware was originally included on a floppy with the delivery of every Magnum. If you don't have the floppy anymore you can download it via anonymous ftp from ftp://oss.sgi.com/pub/linux/mips/misc/magnum−4000.

It is possible to reconfigure the M700 for headless operation by setting the firmware environment variables ConsoleIn and ConsoleOut to multi()serial(0)term(). Also, try the command *listdev* which will show the available ARC devices.

In some cases, like where the G364 graphics card is missing but the console is still configured to use normal graphics, it will be necessary to set the configuration jumper JP2 on the board. After the next reset, the machine will reboot with the console on COM2.

MIPS Magnum 4000SC

The MIPS Magnum 4000SC is the same as a Magnum 4000 (see above) with the exception that it uses an R4000SC CPU.

NEC machines

The NEC uniprocessor machines are OEM *Acer PICA* systems, see that section for details. The SMP systems are different from that. The Linux/MIPS developers have no technical documentation as necessary to write an OS. As long as that does not change, this will pretty much stay a show− stopper, preventing a port to NEC's SMP systems.

Netpower 100

The *Netpower 100* is apparently an *Acer PICA* in disguise. It should therefore be supported but this is untested. If there is a problem then it is probably the machine detection.

Nintendo 64

The *Nintendo 64* is R4300−based game console with 4MB RAM. Its graphics chips were developed by Silicon Graphics for Nintendo. Right now this port has pipe dream status and will continue to be in that state until Nintendo decides to publish the necessary technical information. The question remains as to whether porting the Linux/MIPS code to this platform is a good idea.

Silicon Graphics Challenge S

This machine is very similar to the Indy, the differences are that it doesn't have a keyboard or graphics card, but has an additional SCSI WD33C95−based adapter. This WD33C95 hostadapter is currently not supported.

Silicon Graphics Indigo

This machine is only being mentioned here because people have occasionally confused it with Indys or the Indigo 2. The Indigo is a different R3000−based architecture however, and is yet unsupported.

Silicon Graphics Indigo2

This machine is the successor to the Indigo and is very similar to the Indy. It is now supported, but is lacking in several areas. You will have to use serial console. If you have an Indigo2 and still want to run Linux on it, contact either Florian Lohoff (flo@rfc822.org) or Klaus Naumann (spock@mgnet.de) .

Silicon Graphics Onyx 2

The Onyx 2 is basically an Origin 2000 with additional graphics hardware. As of now, writing Linux support for the graphics hardware has not yet been done. Aside from that, Linux should run just as well as on a normal, headless Origin 2000 configuration.

Silicon Graphics Power Series

This is a very old series of R3000 SMP systems. There is no hardware documentation for these machines, few of them even exist anymore, and the hardware is weird. In short, the chances that Linux will ever run on them are approximating zero. Not that we want to discourage any takers ...

SNI RM200C

In contrast to the RM200 (see below), this machine has EISA and PCI slots. The RM200 is supported with the exception of the availability of the onboard NCR53c810A SCSI controller.

SNI RM200

If your machine has both EISA and PCI slots, then it is an RM200C (please see above). Due to the slight architectural differences of the RM200 and the RM200C, this machine isn't currently supported in the official sources. Michael Engel (engel@numerik.math.uni−siegen.de) has managed to get his RM200 working partially, but the patches haven't yet been included in the official Linux/MIPS sources.

SNI RM300C

The RM300 is technically very similar to the RM200C. It should be supported by the current Linux kernel, but we haven't yet received any reports.

SNI RM400

The RM400 isn't supported.

SNI RW320

This machine is a OEM variant of the SGI Indigo and therefore also unsupported.

NEC VR41xx−based platforms

The Linux VR project is porting Linux to devices based on the NEC VR41xx microprocessors. Many of these devices were originally designed to run Windows CE. The project has produced working kernels with basic drivers for the Vadem Clio, Casio E−105, Everex Freestyle, and more. For more information please see http://linux−vr.org/.

Toshiba TMPR39xx/Philips PR31700 platforms

Similar to the VR41xx, devices with these processors were originally intended for running Windows CE. However, there are working kernels with basic drivers for *Sharp Mobilon* and the *Compaq C−Series*. Support for more devices is under construction. The code is part of the Linux VR project and as such more information can be found at http://linux−vr.org/.

8.4 Hardware we're never going to support

IBM RS6000

As the name says, these are IBM machines which are based on the RS6000 processor series, and, as such, they're not the subject of the Linux/MIPS project. People frequently confuse the IBM RS6000 with the MIPS R6000 architecture. However, the Linux/PPC project might support these machines. Checkout http://www.penguinppc.org/ for further information.

VaxStation

As the name already implies, this machine is a member of Digital Equipment's VAX family. It's mentioned here because people often confuse it with Digital's MIPS−based DECstation family due to the similar type numbers. These two families of architectures share little technical similarities. Unfortunately, the VaxStation, like the entire VAX family, is currently unsupported.

SGI VisPC

This is actually an x86−based system, therefore not covered by this FAQ. There is some limited Linux support available for the older Visual Workstations. The current series of Visual Workstations is an officially supported SGI product. Please see http://oss.sgi.com and http://www.sgi.com for more information.

Motorola 68k−based machines like the Iris 3000

These are *very* old machines, more than ten years old by now. As these machines are not based on MIPS processors, and therefore not supported by the Linux/MIPS project, this document is the wrong place to search for information.

---

9. Supported CPUs

9.1 MIPS32 architecture

All CPUs and cores that conform to the MIPS32 specification, including the MIPS 4K series, Alchemy Au1000/1500.

9.2 MIPS64 architecture

All CPUs and cores that conform to the MIPS64 specification, including the MIPS 5K and 10K series, Sibyte SB1 core / SB1250 SOC.

9.3 R2000, R3000 family

Linux supports the R2000, R3000 family and many processors that were derived from these the two original MIPS processors such as the R3081.

9.4 R4000, R5000 and RM7000 family

Linux supports many of the members of the R4000 family. Currently, these are: R4000PC, R4400PC, R4300, R4600, R4700, R5000, R5230, R5260, RM7000. The list is growing permanently.

Not supported are the R4000MC and R4400MC CPUs (that is multiprocessor systems), as well as R5000 systems with a CPU−controlled second level cache. This means that the cache is controlled by the R5000 itself, in contrast to some external cache controller. The difference is important because, unlike other systems, especially PCs, on MIPS the cache is architecturally visible and needs to be controlled by software.

Special credit goes to Ulf Carlsson (ulfc@engr.sgi.com) who provided the CPU module for debugging the R4000SC / R4400SC support.

9.5 R6000

Sometimes people confuse the R6000, a MIPS processor, with RS6000, a series of workstations made by IBM. So, if you're reading this in hope of finding out more about Linux on IBM machines, then you're reading the wrong document.

The R6000 is currently not supported. It is a 32−bit MIPS ISA 2 processor; apretty interesting and weird piece of silicon. It was developed and produced by a company named *BIT Technology*. Later, NEC took over the semiconductor production. It was built using ECL technology, the same technology that was, and still is, being used to build extremely fast chips like those used in some Cray computers. The processor had its TLB implemented as part of the last couple of lines of the external primary cache, a technology called *TLB slice*. That means its MMU is substantially different from those of the R3000 or R4000 series, which is also one of the reasons why the processor isn't supported.

9.6 R8000

The R8000 is currently unsupported partly because this processor is relatively rare and has only been used in a few SGI machines, and partly because the Linux/MIPS developers don't have such a machine.

The R8000 is a pretty interesting piece of silicon. Unlike the other members of the MIPS family it is a set of seven chips. It's cache and TLB architecture are pretty different from the other members of the MIPS family. It was born as a quick hack to get the floating point crown back to Silicon Graphics before the R10000 is finished.

9.7 R10000

The R10000 is supported as part of the mips64 kernel which currently is supported on the IP22 (SGI Indy, Challenge S and Indigo 2) and Origin.

Due to the very hard−to−handle way this processor works in non−cachecoherent systems, it will probably be some time until we support this processor in such systems. As of today, these systems are the SGI O2 and Indigo

9.8 Processors without TLB

For embedded purposes, there are special derivates of the above CPU available which often lack a full TLB. We don't support those types nor should you ever expect such support to be added.

Hackers may want to take a look at a Linux subset named Microcontroller Linux, or short, ucLinux. This would be supportable on TLB−less processors. Given the little difference between CPU types with and without TLB, we still recommend that you choose a processor with TLB. It's going to save you a lot of engineering.

9.9 Processors with partial or no FPU

Linux/MIPS version 2.4 and later feature a full FPU emulation and therefore can support these processors while maintaining the full binary compatibility to fpu−full versions.

---

10. Technical FAQ

10.1 Installation of Linux/MIPS and common problems.

NFS booting fails.

Usually, the reason for this is that people have unpacked the tar archive under IRIX, not Linux. Since the representation of device files over NFS is not standardized between various Unices, this fails. The symptom is that the system dies with the error message ``Warning: unable to open an initial console.'' right after mounting the NFS filesystem.

For now, the workaround is to use a Linux system (doesn't need to be MIPS) to unpack the installation archive onto the NFS server. The NFS server itself may be any type of UNIX.

Self−compiled kernels crash when booting.

When I build my own kernel, it crashes. On an Indy the crash message looks like the following (the same problem hits other machines as well but may look completely different):

```
Exception: <vector=UTLB Miss>
Status register: 0x300004803<CU1,CU0,IM4,IPL=???,MODE=KERNEL,EXL,IE>
Cause register: 0x8008<CE=0,IP8,EXC=RMISS>
Exception PC: 0x881385cc, Exception RA: 0x88002614
exception, bad address: 0x47c4
Local I/O interrupt register 1: 0x80 <VR/GIO2>
Saved user regs in hex (&gpda 0xa8740e48, &_regs 0xa8741048):
     arg: 7 8bfff938 8bfffc4d 880025dc
     tmp: 8818c14c 8818c14c 10 881510c4 14 8bfad9e0 0 48
     sve: 8bfdf3e8 8bfffc40 8bfb2720 8bfff938 a8747420 9fc56394 0 9fc56394
     t8 48 t9 8bfffee66 at 1 v0 0 v1 8bfff890 k1 bad11bad
     gp 881dfd90 fp 9fc4be88 sp 8bfff8b8 ra 88002614

PANIC: Unexpected exception
```

This problem is caused by a still unfixed bug in Binutils newer than version 2.7. As a workaround, change the following line in arch/mips/Makefile from:

```
LINKFLAGS       = -static -N
```

to:

```
LINKFLAGS       = -static
```

Booting the kernel on the Indy fails with PROM error messages

```
>> boot bootp()/vmlinux
73264+592+11520+331680+27848d+3628+5792 entry: 0x8df9a960
Setting $netaddres to 192.168.1.5 (from server deadmoon)
Obtaining /vmlinux from server deadmoon

Cannot load bootp()/vmlinux
Illegal f_magic number 0x7f45, expected MIPSELMAGIC or MIPSEBMAGIC.
```

This problem only happens for Indys with very old PROM versions which cannot handle the ELF binary format which Linux uses. A solution for this problem is in the works.

Where can I get the little endian firmware for my SNI?

SNI's system can be operated in both big and little endian modes. At this time, Linux/MIPS only supports the little endian firmware. This is somewhat unlucky since SNI hasn't shipped that firmware for quite some time, since they dropped Windows NT.

When running in big endian mode, the firmware looks similar to an SGI Indy which is already supported, therefore fixing the SNI support will be relatively easy. Interested hackers should contact Ralf Bächle (ralf@gnu.org).

ld dies with signal 6

```
collect2: ld terminated with signal 6 [Aborted]
```

This is a known bug in older binutils versions. You will have to upgrade to binutils 2.8.1 plus very current patches.

Missing ELF support in some PROM versions

Old PROM versions don't know about the ELF binary format which the Linux kernel normally uses, so Linux cannot boot directly. This results in error messages similar to this one:

```
>> boot -f linux root=/dev/sda1

Cannot load scsi(0)disk(1)rdisk(0)partition(8)/linux.
Illegal f_magic number 0x7f45, expected MIPSELMAGIC or MIPSEBMAGIC.
Unable to load linux: ``linux'' is not a valid file to boot.
>>
```

The preferable solution for this is of course a PROM upgrade but that isn't available for all systems.

For systems which still have the sash of IRIX 5 installed it is alternativly possible use Sash to boot the kernel. Sash knows how to load ELF binaries and doesn't care if it's an IRIX or Linux kernel. Simply type ``Sash'' to the PROM monitor. You'll get another shell prompt, this time from Sash. Now launch Linux as usual.

Sash can read EFS or XFS filesystems or read the kernel from BOOTP / TFTP.

Using the elf2ecoff tool that is shipping with the kernel source you can convert an ELF binary into ECOFF. Or when building a kernel just run the ``make vmlinux.ecoff'' which will produce an ECOFF kernel.

My machine doesn't download the kernel when I try to netboot

Your machine is replying to the BOOTP packets (you may verify this using a packet sniffer like tcpdump or ethereal), but doesn't download the kernel from your BOOTP server. This happens if your boot server is running a kernel of the 2.3 series or higher. The problem may be circumvented by doing a "echo 1 > /proc/sys/net/ipv4/ip_no_pmtu_disc" as root on your boot server.

The kernel download from the TFTP server stops and times out

This may happen if the TFTP server is using a local port number of 32768 or higher which usually happens if the TFTP server is running Linux 2.3 or higher. This problem may be circumvented by doing a "echo 2048 32767 > /proc/sys/net/ipv4/ip_local_port_range" on the server.

Bug in DHCP version 2

When using DHCP version 2 you might see the following problem: Your machines receives it's BOOTP reply 3 times but refuses to start TFTP. You can fix this by doing a "unsetenv netaddr" in the PROM command monitor before you boot your system. DHCP version 3 fixes that problem.

When booting I get: Warning: unable to open an initial console.

This problem has two possible solutions. First make sure you actually have a driver for the console of your system configured. If this is the case and the problem persists you probably got victim of a widespread bug in Linux distributions and root filesystems out there. The console of a Linux systems should be a character

device of major id 5, minor 1 with permissions of 622 and owned by user and group root. If that's not the case, cd to the root of the filesystem and execute the following commands as root:

```
rm -f dev/console
mknod --mode=622 dev/console
```

You can also do this on a NFS root filesystem, even on the NFS server itself. However note that the major and minor ids are changed by NFS, therefore you must do this from a Linux system even if it's only a NFS client or the major / minor ID might be wrong when your Linux client boots from it.

Is IRIX required for installation on SGI systems?

Various descriptions of the installation procedure use IRIX in order to partition disks. This was required at the time of their writing as there were no native partiting tools available. Now disks can be partitioned using the IRIX disklabel mode which can be selected in the expert menu of newer fdisk versions. The volume header can be manipulated using dvhtool. Note dvhtool usage is different from IRIX.

IRIX as secondary operating systems can still be handy as it may reduce the need to fiddle with ramdisks or nfs−root during installation. Just one word of warning though: Be careful to not point IRIX fx(8) to disks that don't don't contain an IRIX disklabel if you want to retain the content − IRIX may *damage* the content of that disk without asking!

Can IRIX and Linux be installed on the same system

Yes. Just make sure you read the warning about IRIX's fx(8) in above paragraph.

Insmod complains about the _gp_disp symbol being undefined

_gp_disp is a magic symbol used with PIC code on MIPS. Be happy, this error message saved you from crashing your system. You should use the same compiler options to compile a kernel module as the kernel makefiles do. In particular the options *−mno−pic −mno−abicalls −G 0* are important.

Serial console on SGI machines

Make sure that the kernel you're using includes the appropriate driver for a serial interface and serial console. Set the *console* ARC environment variable to either the value *d1*, or *d2* for Indy and Challenge S depending on which serial interface you're going to use as the console.

If you have the problem that all kernel messages appear on the serial console on boot−up, but everything is missing from the point when init starts, then you probably have the wrong setup for your /dev/console. You can find more information about this in the Linux kernel source documentation which is in /usr/src/linux/Documentation/serial−console.txt if you have the kernel source installed.

Strange amounts of available memory on SGI

On bootup, the kernel on the Indy will report available memory with a message like:

```
Memory: 27976k/163372k available (1220k kernel code, 2324k data)
```

The large difference between the first pair of numbers is caused by a 128MB area in the Indy's memory address space which mirrors up to the first 128MB of memory. The difference between the two numbers will always be about 128MB and does not indicate a problem of any kind. Kernels since 2.3.23 don't count this 128MB gap any more.

Indy PROM related problems

Several people have reported these problems with their machines after upgrading them typically from surplus parts. There are several PROM versions for the Indy available. Machines with old PROM versions which have been upgraded to newer CPU variants, like a R4600SC or R5000SC module, can crash during the self test with an error message like:

```
Exception: <vector=Normal>
Status register: 0x30004803<CU1,CU0,IM7,IM4,IPL=???,MODE=KERNEL,EXL,IE>
Cause register: 0x4000<CE=0,IP7,EXC=INT>
Exception PC: 0xbfc0b598
Interrupt exception
CPU Parity Error Interrupt
Local I/O interrupt register 1: 0x80 <VR/GIO2>
CPU parity error register: 0x80b<B0,B1,B3,SYSAD_PAR>
CPU parity error: address: 0x1fc0b598
NESTED EXCEPTION #1 at EPC: 9fc3df00; first exception at PC: bfc0b598
```

In that case, you'll have to upgrade your machine's PROM to a newer version, or go back to an older CPU version (usually R4000SC or R4400SC modules should work). Just to be clear, this is a problem which is unrelated to Linux, it is only mentioned here because several Linux users have asked about it.

Why is so much memory reserved on my Indy?

On bootup, the `Memory: ...' message on an Indy says that there is 128MB of RAM reserved. That is ok. Just like the PC architecture has a gap in its memory address space between 640KB and 1024KB, the Indy has a 128MB−sized area in its memory map where the first 128MB of its memory is mirrored. Linux knows about it and just ignores that memory, and thus this message.

10.2 Milo

Milo is the boot loader used to boot the little endian MIPS systems with ARC firmware, currently the Jazz family and the SNI RM 200. While Milo uses the same name and has a similar purpose to the Alpha version of Milo, these two Milos have nothing else in common. They were developed by different people, don't share any code, and work on different hardware platforms. The fact that both have the same name is just a kind of historic ``accident''.

The need for Milo has been eliminated for all ARC platforms except the RM200C due to it's unusual firmware behavior. On all other platforms an ECOFF or often on more modern firmware also an ELF kernel can be started directly without the need for Milo or an equivalent. On the RM200C Milo 0.27.1 is still required to boot the kernel.

Building Milo

The building procedure of Milo is described, in detail, in the README files in the Milo package. Since Milo

has some dependencies to kernel header files which have changed over time, Milo often cannot be built easily. However, the Milo distribution includes binaries for both Milo and Pandora. Building Milo is not trivial; unless you want to modify Milo yourself the urgent recommendation is to use the binaries shipping in the Milo tarball.

Pandora

Pandora is a simple debugger which was primarily developed in order to analyze undocumented systems. Pandora includes a disassembler, memory dump functions, and more. If you only want to use Linux, then there is no need to install Pandora, despite its small size.

10.3 Loadable Modules

Using modules on Linux/MIPS is quite easy. It should work as expected for people who have used the feature on other Linux systems. If you want to run a module−based system, then you should have at least kernel version 980919, and modutils newer than version 2.1.121 installed. Older versions won't work.

10.4 How do I set up a cross−compiler?

Available binaries

The easiest way to setup a cross−compiler is to just download the binaries. For Linux/i386 hosts and big endian targets, these are the packages:

```
binutils−mips−linux−2.8.1−1.i386.rpm
egcs−c++−mips−linux−1.1.2−2.i386.rpm
egcs−g77−mips−linux−1.1.2−2.i386.rpm
egcs−libstdc++−mips−linux−2.9.0−2.i386.rpm
egcs−mips−linux−1.1.2−2.i386.rpm
egcs−objc−mips−linux−1.1.2−2.i386.rpm
```

And this is the list of packages for little endian targets:

```
binutils−mipsel−linux−2.8.1−1.i386.rpm
egcs−c++−mipsel−linux−1.1.2−2.i386.rpm
egcs−g77−mipsel−linux−1.1.2−2.i386.rpm
egcs−libstdc++−mipsel−linux−2.9.0−2.i386.rpm
egcs−mipsel−linux−1.1.2−2.i386.rpm
egcs−objc−mipsel−linux−1.1.2−2.i386.rpm
```

For 64−bit MIPS kernels, there is only one package available right now:

```
egcs−mips64−linux−1.1.2−2.i386.rpm
```

This compiler is only available in the big endian flavor as there currently is no little endian machine supported by the 64−bit kernel. A little endian version of the compiler will be provided as soon as there is demand for one.

It's not necessary that you install all of these packages as most people can just omit the C++, Objective C and Fortran 77 compilers. The Intel binaries have been linked against GNU libc 2.1, so you may have to install that as well when upgrading.

Recommended compiler versions

Compilers older than egcs 1.1.2 are no longer supported for compiling kernels due to bugs in the generated code. At this time, we still recommend binutils 2.8.1 despite their age.

Building your own cross−compiler

First of all, go and download the following source packages:

- binutils−2.8.1.tar.gz
- egcs−1.1.2.tar.gz
- glibc−2.0.6.tar.gz
- glibc−crypt−2.0.6.tar.gz
- glibc−localedata−2.0.6.tar.gz
- glibc−linuxthreads−2.0.6.tar.gz

You can obtain these files from your favorite GNU archive or oss.sgi.com. Furthermore, you'll need patches. The unbundled patch files aren't always up−to−date and additional, not MIPS−specific, patches may be required for building. Note that the unbundled patch files also use a different revision numbering and it is therefore recommended that you obtain the source and patches from the RPM packages distributed on oss.sgi.com.

Those are the currently recommended versions. Older versions may or may not be working. If you're trying to use older versions, please don't send bug reports because we don't care. When installing, please install things in the order of binutils, egcs, then glibc. Unless you have older versions already installed, changing the order *will* fail.

Disk space requirements

For the installation, you'll have to choose a directory where the files will be installed. I'll refer to that directory below with <prefix>. To avoid a particular problem, it's best to use the same value for <prefix> as your native gcc. For example, if your gcc is installed in /usr/bin/gcc, then choose /usr for <prefix>. You must use the same <prefix> value for all the packages that you're going to install.

During compilation, you'll need about 31MB disk space for binutils. For installation, you'll need 7MB disk space on <prefix>'s partition. Building egcs requires 71MB, and installation 14MB. GNU libc requires 149MB disk space during compilation, and 33MB for installation. Note, these numbers are just a guideline and may differ significantly for different processor and operating system architectures or compiler options.

Byte order

One of the special features of the MIPS architecture is that all processors except the R8000 can be configured to run either in big or in little endian mode. Byte order means the way the processor stores multibyte numbers in memory. Big endian machines store the byte with the highest value digits at the lowest address while little endian machines store it at the highest address. Think of it as writing multi−digit numbers from left to right or vice versa.

In order to setup your cross−compiler correctly, you have to know the byte order of the cross−compiler target. If you don't already know, check the section Hardware Platforms for your machine's byte order.

Configuration names

Many of the packages based on autoconf support many different architectures and operating systems. In order to differentiate between these many configurations, names are constructed with <cpu>−<company>−<os>, or even <cpu>−<company>−<kernel>−<os>. Expressed this way, the configuration names of Linux/MIPS are: mips−unknown−linux−gnu for big endian targets, or mipsel−unknown−linux−gnu for little endian targets. These names are a bit long and are allowed to be abbreviated to mips−linux or mipsel−linux. You *must* use the same configuration name for all packages that comprise your cross−compilation environment. Also, while other names, like mips−sni−linux or mipsel−sni−linux, are legal configuration names, use mips−linux or mipsel−linux instead. These are the configuration names known to other packages, like the Linux kernel sources, and they would otherwise have to be changed for cross−compilation.

I'll refer to the target configuration name below with <target>.

Installation of GNU Binutils.

This is the first and simplest part (at least as long as you're trying to install on any halfway−sane UNIX flavour). Just cd into a directory with enough free space and do the following:

```
gzip −cd binutils−<version>.tar.gz | tar xf −
cd binutils−<version>
patch −p1 < ../binutils−<version>−mips.patch
./configure −−prefix=<prefix> −−target=<target>
make CFLAGS=−O2
make install
```

This usually works correctly. However, certain machines using GCC 2.7.x as compiler are known to dump core. This is a known bug in GCC and can be fixed by upgrading the host compiler to GCC 2.8.1 or better.

Assert.h

Some people have an old assert.h header file installed, probably leftover from an old cross−compiler installation. This file may cause autoconf scripts to fail silently. Assert.h was never necessary and was only installed because of a bug in older GCC versions. Check to see if the file <prefix>/<target>/include/assert.h exists in your installation. If so, just delete the it − it should never have been installed for any version of the cross−compiler and will cause trouble.

Installing the kernel sources

Installing the kernel sources is simple. Just place them into some directory of your choice and configure them. Configuring them is necessary so that files which are generated by the procedure will be installed. Make sure you enable CONFIG_CROSSCOMPILE near the end of the configuration process. The only problem you may run into is that you may need to install some required GNU programs like bash or have to override the manufacturer−provided versions of programs by placing the GNU versions earlier in the PATH variable. Now, go to the directory <prefix>/<target>/include and create two symbolic links named asm and linux pointing to include/asm rsp. include/linux within your just installed and configured kernel sources. These are necessary such that the necessary header files will be found during the next step.

First installation of egcs

Now the pain begins. There is a so−called bootstrap problem. In our case, this means that the installation process of egcs needs an already installed glibc, but we cannot compile glibc because we don't have a working cross−compiler yet. Luckily, you'll only have to go through this once when you install a cross−compiler for the first time. Later, when you already have glibc installed, things will be much smoother. So now do:

```
gzip −cd egcs−1.1.2.tar.gz | tar xf −
cd egcs−<version>
patch −p1 < ../egcs−1.1.2−mips.patch
./configure −−prefix=<prefix> −−with−newlib −−target=<target>
make SUBDIRS="libiberty texinfo gcc" ALL_TARGET_MODULES= \
        CONFIGURE_TARGET_MODULES= INSTALL_TARGET_MODULES= LANGUAGES="c"
```

Note that we deliberately don't build gcov, protoize, unprotoize, and the libraries. Gcov doesn't make sense in a cross−compiler environment, and protoize and unprotoize might even overwrite your native programs − this is a bug in the gcc makefiles. Finally, we cannot build the libraries because we don't have glibc installed yet. If everything went successfully, install with:

```
make SUBDIRS="libiberty texinfo gcc" INSTALL_TARGET_MODULES= \
        LANGUAGES="c" install
```

If you only want the cross−compiler for building the kernel, you're done. Cross−compiling libc is only required to be able to compile user applications.

Test what you've done so far

Just to make sure that what you've done so far is actually working, you may now try to compile the kernel. Cd to the MIPS kernel's sources and type ``make clean; make dep; make''. If everything went ok do ``make clean'' once more to clean the sources.

Installing GNU libc

Note: Building glibc 2.0.6 using a compiler newer than egcs 1.0.3a is not recommended due to binary compatibility problems which may hit certain software. It's recommended that you either use egcs 1.0.3a or use the files from a published binary package. Crosscompiling GNU libc is always only the second best solution as certain parts of it will not be compiled when crosscompiling. A proper solution will be documented here as soon as it is available and believed to be stable. With this warning given, here's the recipe:

```
gzip −cd glibc−2.0.6.tar.gz | tar xf −
cd glibc−2.0.6
gzip −cd glibc−crypt−2.0.6.tar.gz | tar xf −
gzip −cd glibc−localedata−2.0.6.tar.gz | tar xf −
gzip −cd glibc−linuxthreads−2.0.6.tar.gz | tar xf −
patch −p1 < ../glibc−2.0.6−mips.patch
mkdir build
cd build
CC=<target>−gcc BUILD_CC=gcc AR=<target>−ar RANLIB=<target>−ranlib \
      ../configure −−prefix=/usr −−host=<target> \
      −−enable−add−ons=crypt,linuxthreads,localedata −−enable−profile
make
```

You now have a compiled GNU libc which still needs to be installed. Do *not* just type make install. That would overwrite your host system's files with Linux/MIPS−specific files with disastrous effects. Instead, install GNU libc into some other arbitrary directory <somedir> from which we'll move the parts we need for cross−compilation into the actual target directory:

```
make install_root=<somedir> install
```

Now cd into <somedir> and finally install GNU libc manually:

```
cd usr/include
find . −print | cpio −pumd <prefix>/<target>/include
cd ../../lib
find . −print | cpio −pumd <prefix>/<target>/lib
cd ../usr/lib
find . −print | cpio −pumd <prefix>/<target>/lib
```

GNU libc also contains extensive online documentation. Your system might already have a version of this documentation installed, so if you don't want to install the info pages, which will save you a less than a megabyte, or already have them installed, skip the next step:

```
cd ../info
gzip −9 *.info*
find . −name \*.info\* −print | cpio −pumd <prefix>/info
```

If you're not bootstrapping, your installation is now finished.

Building egcs again

The first attempt of building egcs was stopped by lack of a GNU libc. Since we now have libc installed we can rebuild egcs but this time as complete as a cross−compiler installation can be:

```
gzip −cd egcs−<version>.tar.gz | tar xf −
cd egcs−<version>
patch −p1 < ../egcs−1.1.2−mips.patch
./configure −−prefix=<prefix> −−target=<target>
make LANGUAGES="c c++ objective−c f77"
```

As you can see, the procedure is the same as the first time, with the exception that we dropped the −−with−newlib option. This option was necessary to avoid the libgcc build breaking due to the lack of libc. Now install with:

```
make LANGUAGES="c c++ objective−c f77" install
```

You're almost finished. If you think you don't need the Objective C or F77 compilers, you can omit them from above commands. Each will save you about 3MB. Do not build gcov, protoize, or unprotoize.

Should I build the C++, Objective C or F77 compilers?

The answer to this question largely depends on your use of your cross−compiler environment. If you only intend to rebuild the Linux kernel, then you have no need for the full blown setup and can safely omit the Objective C and F77 compilers. You must, however, build the C++ compiler, because building the libraries included with the egcs distribution requires C++.

How about float.h?

The installation of float.h is no longer necessary. Since about egcs 1.0.3a, a proper float.h header file will automatically be generated and installed.

Known problem when cross−compiling

IRIX crashes

Origin 200 running IRIX 6.5.1 may crash when running ``make depend'' on the Linux kernel sources. IRIX 6.5 on Indy and IRIX 6.5.4 on Origin 200 are known to work.

Resource limits on System V based hosts

Typical System V−based Unices, like IRIX or Solaris, have limits for the maximum number of arguments to be passed to a child process which may be exceeded when cross−compiling some software like the Linux kernel or GNU libc. For IRIX systems, the maximum length of the argument list defaults to 20KB, while Linux defaults to at least 128KB. This size can be modified by the command ``systune ncargs 131072'' as root.

GDB

Building GDB as cross−debugger is only of interest to kernel developers. For them, GDB may be a life saver. Such a remote debugging setup always consists of two parts: the remote debugger GDB running on one machine, and the target machine running the Linux/MIPS kernel being debugged. The machines are typically interconnected with a serial line. The target machine's kernel needs to be equipped with a ``debugging stub'' which communicates with the GDB host machine using the remote serial protocol.

Depending on the target's architecture, you may have to implement the debugging stub yourself. In general, you'll only have to write very simple routines for the serial line. The task is further simplified by the fact that most machines are using similar serial hardware, typically based on the 8250, 16450 or derivatives.

10.5 Compiling the kernel

Choosing a processor type

R2000, R3000 family

For these processors just select the R3000 option. A kernel built for this option will not run on any other processors than R2000 and R3000 family members.

R4000, R5000 family

With the exception of the Nevada family these processors are all fully compatible with rescpect to the kernel. Choose the option which matches your processor best for optimal performance.

R6000

Linux currently doesn't support the R6000 so this paragraph is entirely theoretical. The R6000 has it's own, rather unique if not odd cache and MMU architecture; it also requires alot of workarounds as it's a quite broken piece of silicon. Therefore a R6000 kernel will not work on any other processor nor will a kernel for another processor work on the R6000.

Nevada

The Nevada nickname stands for the QED 5230, 5231, R5260, R5261, R5270 etc. family of CPUs. It enables the use of additional instructions which are not supported on other processors therefore you only should choose this opition if you indeed have one of these processors. If you're not sure configure for R4x00 or R5000 (see above) and

SB1

Choose this option only for the Sibyte SB1 processor. A kernel built for this processor will not work on any other processor type nor vice versa. Being a truly bleeding edge OS Linux supports this processor even though silicon doesn't even exist yet.

R10000

Choose this option if you want to run Linux on a R10000, R12000 or R14000 system. A kernel built with this option will not work on R4000 or R5000 family processors.

MIPS32

Choose this option if you want to run Linux on a member of the MIPS32 family.

Compatible options

The kernel configuration process doesn't make a too strong attempt at making wrong configuration impossible. So for example an SGI Indy may never have a framebuffer, yet it's possible to enable it which later on will result in a compile error. This situation will improve in the future when CML2 will be the standard kernel configuration language; for 2.2 and 2.4 you still will have to care of your steps yourself.

Crosscompilation

The kernel has been carefully developped to ensure crosscompilation on a non−MIPS system is possible. Once you've managed to get around the cliff of setting up a crosscompiler crosscompiling is easy. To do so you have two options. First you can pass CROSS_COMPILE=<target>− (note the trailing dash) as an additional argument to your make invocations where you choose one of mips−linux, mipsel−linux, mips64−linux or mips64el−linux depending if your target is big or little endian, 32−bit or 64−bit. An alternate way is setting the CONFIG_CROSSCOMPILE configuration option. The kernel will then automatically

choose the right value for CROSS_COMPILE which will keep make command lines a bit simpler.

32−bit vs. 64−bit

By default the Linux/MIPS kernel source tree is configured to build a 32−bit target. If you want to build a 64−bit kernel you'll have pass the additional ARCH=mips64 argument to all you make invocations.

---

11. Documentation

11.1 Getting this information as a single document

You can download this document in various formats:

- The HTML version http://oss.sgi.com/mips/mips−howto.html
- The text version http://oss.sgi.com/mips/mips−howto.txt
- The Postscript version http://oss.sgi.com/mips/mips−howto.ps
- The Linux−Doc SGML version. http://oss.sgi.com/mips/mips−howto.sgml

This FAQ is also available as SGML source code via anonymous CVS from oss.sgi.com. The archive also has a Makefile which will translate it into various formats. An ASCII version is regularly being posted via *comp.os.linux.answers* and the other Linux HOWTO channels.

Updates for this document should be sent as unified diffs against the SGML version to Ralf Bächle (ralf@gnu.org). Please don't updates in any other form as that will make maintenance significantly more difficult.

11.2 See MIPS Run

Author Dominic Sweetman, Publisher Morgan Kaufmann, ISBN 1−55860−410−3.

This is intended as a pretty comprehensive guide to programming MIPS, wherever it's different from programming any other 32−bit CPU. It's the first time anyone has tried to write a readable, and comprehensive, explanation and account of the wide range of MIPS CPUs available. It should be very helpful for anyone programming MIPS who isn't insulated by someone else's operating system. Also, the author is a free−unix enthusiast who subscribes to the Linux/MIPS mailing list!

John Hennessey, father of the MIPS architecture, was kind enough to write in the foreword: ``... this book is the best combination of completeness and readability of any book on the MIPS architecture ...";

It includes some context about RISC CPUs, a description of the architecture and instruction set, including the "co−processor 0" instructions used for CPU control; sections on caches, exceptions, memory management, and floating point. There's a detailed assembly language guide, some stuff about porting, and some fairly heavy−duty software examples.

Available from:

- Algorithmics (Europe)
- Morgan Kaufmann (US)

- Amazon USA
- Amazon UK

and from good bookshops anywhere. It's 512 pages and costs around $50 in the US, £34 in the UK.

I'd be inclined to list two other books too, both from Morgan Kaufmann and available from www.mkp.com or any good bookshop:

11.3 The MIPS Programmer's Handbook

Authors Farquhar and Bunce, Publisher Morgan Kaufmann, ISBN 1−55860−297−6.

A readable introduction to the practice of programming MIPS at the low level, by the author of PMON. Strengths: lots of examples; weakness: leaves out some big pieces of the architecture (such as memory management, floating point and advanced caches) because they didn't feature in the LSI ``embedded'' products this book was meant to partner.

11.4 Computer Architecture − A Quantitative Approach

Authors Hennessy & Patterson, Publisher Morgan Kaufmann, ISBN 1−55860−329−8.

The bible of modern computer architecture and a must−read if you want to understand what makes programs run slow or fast. Is it about MIPS? Well, it's mostly about something very *like* MIPS... Its sole defect is its size and weight − but unlike most big books it's worth every page.

11.5 MIPS ABI documentation

The documentation to be found at ftp://www.linux−mips.org/pub/linux/mips/doc/ABI/ defines many of the MIPS specific technical standards like calling conventions, ELF properties, and much more that is being used by Linux/MIPS, including the N32 standard.

11.6 The mips.com site

Under http://www.mips.com/publications there are various PDF documents and data sheets about individual processors and cores.

11.7 The NEC site

NEC Electronics (http://www.necel.com includes complete manuals about their VR41xx processors.

11.8 techpubs.sgi.com

While being very SGI centric http://techpubs.sgi.com has a number of ABI related documents online that also apply to Linux/MIPS.

12. Legal Notices

12.1 Copyright

Except where otherwise specified, the information in this documentation or website is copyright (c) 1998,1999,2000,2001,2002 Ralf Bächle.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being Copyright, with no Front−Cover Texts and with no Back−Cover Texts.

A copy of the GNU Free Documentation License is available on the World Wide Web at http://www.gnu.org/copyleft/fdl.html You can also obtain it by writing to the

```
Free Software Foundation, Inc.
59 Temple Place − Suite 330
Boston, MA 02111−1307
USA
```

12.2 Software Use

Any software contained in or linked to by this documentation (the "Software") is copyrighted work. To use the Software you must comply with the terms of the Software's license agreement. SOFTWARE IS WARRANTED, IF AT ALL, IN ACCORDANCE WITH THE TERMS OF THE LICENSE AGREEMENT. EXCEPT AS SET FORTH IN THE LICENSE AGREEMENT, ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON−INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

12.3 Links to websites

This documentation may contain links to websites which are not under our control. We are not responsible for the content of those sites. The links are available only as a convenience, and the inclusion of any link to such sites does not imply endorsement of those sites.

12.4 Trademarks

Linux is a Registered Trademark of Linus Torvalds.

MIPS is a Registered Trademark of MIPS Technologies, Inc.

12.5 Disclaimer

Note that, as provided in the License, the software on this website is distributed on an "AS IS" basis, with ALL EXPRESS AND IMPLIED WARRANTIES AND CONDITIONS DISCLAIMED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES AND CONDITIONS OF

MERCHANTABILITY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON−INFRINGEMENT.

12.6 Limitation of liability

THE AUTHORS OF THIS WEB SITE SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED AS A RESULT OF USING, MODIFYING, CONTRIBUTING, COPYING, DISTRIBUTING, OR DOWNLOADING THE MATERIALS ON THIS WEBSITE. IN NO EVENT SHALL WE BE LIABLE FOR ANY INDIRECT, PUNITIVE, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGE (INCLUDING LOSS OF BUSINESS, REVENUE, PROFITS, USE, DATA OR OTHER ECONOMIC ADVANTAGE) HOWEVER IT ARISES, WHETHER FOR BREACH OR IN TORT, EVEN IF WE HAVE BEEN PREVIOUSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

YOU HAVE SOLE RESPONSIBILITY FOR ADEQUATE PROTECTION AND BACKUP OF DATA AND/OR EQUIPMENT USED IN CONNECTION WITH THE WEBSITE AND WILL NOT MAKE A CLAIM AGAINST THIS WEB SITE OR ITS AUTHORS FOR LOST DATA, RE−RUN TIME, INACCURATE OUTPUT, WORK DELAYS OR LOST PROFITS RESULTING FROM THE USE OF THE MATERIALS. YOU AGREE TO HOLD US HARMLESS FROM, AND YOU COVENANT NOT TO SUE US FOR, ANY CLAIMS BASED ON USING THE WEBSITE.
