Linux/MIPS HOWTO

Table of Contents

Linux/MIPS HOWTO 1
Ralf Bächle, ralf@gnu.org 1
1. What is Linux/MIPS? 1
2. Getting this FAQ 1
3. What hardware does Linux/MIPS support? 1
4. Linux distributions. 1
5. Linux/MIPS net resources. 1
6. Installation of Linux/MIPS and common problems. 1
7. Milo 2
8. Loadable Modules 2
9. How do I set up a cross−compiler? 2
10. Related Literature 2
1. What is Linux/MIPS? 2
2. Getting this FAQ 3
3. What hardware does Linux/MIPS support? 3
3.1 Hardware Platforms 3
Acer PICA 3
Baget/MIPS series 3
Cobalt Qube and Raq 4
NEC machines 4
NEC VR41xx−based platforms 4
Toshiba TMPR39xx/Philips PR31700 platforms 4
Netpower 100 4
Nintendo 64 4
Silicon Graphics Challenge S 5
Silicon Graphics Indigo 5
Silicon Graphics Indigo2 5
Silicon Graphics Indy 5
Strange amounts of available memory 5
Indy PROM related problems 5
ELF support in old PROM versions 6
Why is so much memory reserved on my Indy? 6
Silicon Graphics Origin 200 and 2000 6
Silicon Graphics Onyx 2 6
Silicon Graphics Power Series 6
Serial console on SGI machines 7
Other Silicon Graphics machines 7
Sony Playstation 7
SNI RM200C 7
SNI RM200 7
SNI RM300C 7
SNI RM400 7
SNI RW320 7
Algorithmics P−4032, P−5064, P−6032 8
DECstation series 8
Mips Magnum 4000 / Olivetti M700−10 9
MIPS Magnum 4000SC 9

Table of Contents

3.2 Processor types....9
- R2000, R3000 family....9
- R6000....10
- R4000 and R5000 family....10
- R8000....10
- R10000....10
- Processors without TLB....11
- Processors with partial or no FPU....11

3.3 Hardware we're never going to support....11
- IBM RS6000....11
- VaxStation....11
- SGI VisPC....11
- Motorola 68k−based machines like the Iris 3000 11

4. Linux distributions....11
4.1 RedHat....12
4.2 Debian....12
4.3 Simple....12
4.4 Other....12
5. Linux/MIPS net resources....12
5.1 Anonymous FTP servers....12
5.2 Anonymous CVS servers....13
5.3 Web servers....13
5.4 Web CVS server....14
5.5 Mailing lists....14
5.6 IRC channel....14
6. Installation of Linux/MIPS and common problems....14
6.1 NFS booting fails....14
6.2 Self−compiled kernels crash when booting....14
6.3 Booting the kernel on the Indy fails with PROM error messages....15
6.4 Where can I get the little endian firmware for my SNI?....15
6.5 ld dies with signal 6....16
6.6 My machine doesn't download the kernel when I try to netboot....16
6.7 Bug in DHCP version 2....16
7. Milo....16
7.1 Building Milo....16
7.2 Pandora....16
8. Loadable Modules....17
9. How do I set up a cross−compiler?....17
9.1 Available binaries....17
9.2 Recommended compiler versions....17
9.3 Building your own cross−compiler....18
9.4 Disk space requirements....18
9.5 Byte order....18
9.6 Configuration names....18
9.7 Installation of GNU Binutils....19
9.8 Assert.h....19
9.9 Installing the kernel sources....19

Table of Contents

9.10 First installation of egcs....19
9.11 Test what you've done so far....20
9.12 Installing GNU libc....20
9.13 Building egcs again....21
9.14 Should I build the C++, Objective C or F77 compilers?....21
9.15 How about float.h?....22
9.16 Known problem when cross−compiling....22
IRIX crashes....22
Resource limits on System V based hosts....22
9.17 GDB....22
10. Related Literature....22
10.1 See MIPS Run....22
10.2 The MIPS Programmer's Handbook....23
10.3 Computer Architecture − A Quantitative Approach....23
10.4 UNIX System V ABI MIPS Processor Supplement....23
10.5 The mips.com site....23
10.6 The NEC site....23

Linux/MIPS HOWTO

Ralf Bächle, `ralf@gnu.org`

January 8, 2001

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.

---

1. What is Linux/MIPS?

2. Getting this FAQ

3. What hardware does Linux/MIPS support?

- 3.1 Hardware Platforms
- 3.2 Processor types
- 3.3 Hardware we're never going to support

4. Linux distributions.

- 4.1 RedHat
- 4.2 Debian
- 4.3 Simple
- 4.4 Other

5. Linux/MIPS net resources.

- 5.1 Anonymous FTP servers.
- 5.2 Anonymous CVS servers.
- 5.3 Web servers.
- 5.4 Web CVS server.
- 5.5 Mailing lists.
- 5.6 IRC channel.

6. Installation of Linux/MIPS and common problems.

- 6.1 NFS booting fails.
- 6.2 Self−compiled kernels crash when booting.
- 6.3 Booting the kernel on the Indy fails with PROM error messages
- 6.4 Where can I get the little endian firmware for my SNI?
- 6.5 ld dies with signal 6
- 6.6 My machine doesn't download the kernel when I try to netboot

- 6.7 Bug in DHCP version 2

7. Milo

- 7.1 Building Milo
- 7.2 Pandora

8. Loadable Modules

9. How do I set up a cross−compiler?

- 9.1 Available binaries
- 9.2 Recommended compiler versions
- 9.3 Building your own cross−compiler
- 9.4 Disk space requirements
- 9.5 Byte order
- 9.6 Configuration names
- 9.7 Installation of GNU Binutils.
- 9.8 Assert.h
- 9.9 Installing the kernel sources
- 9.10 First installation of egcs
- 9.11 Test what you've done so far
- 9.12 Installing GNU libc
- 9.13 Building egcs again
- 9.14 Should I build the C++, Objective C or F77 compilers?
- 9.15 How about float.h?
- 9.16 Known problem when cross−compiling
- 9.17 GDB

10. Related Literature

- 10.1 See MIPS Run
- 10.2 The MIPS Programmer's Handbook
- 10.3 Computer Architecture − A Quantitative Approach
- 10.4 UNIX System V ABI MIPS Processor Supplement
- 10.5 The mips.com site
- 10.6 The NEC site

---

1. What is Linux/MIPS?

Linux/MIPS is a port of the widespread UNIX clone Linux to the MIPS architecture. Linux/MIPS is running on a large number of technically very different systems ranging from small embedded systems and servers to large desktop machines and servers that, at least at the time when they were introduced into the market, were the best of their class.

Linux/MIPS advantages over other operating systems at this time are

- The entire Linux system consists only of Free Software.
- Excellent Price/Performance ratio.
- Availability of large amounts of software of which a large part again is Free Software.
- Binary compatibility across a growing number of platforms.
- Small footprint making Linux/MIPS suitable for many embedded systems.

In short, Linux has been designed and ships with Fahrvergnügen. However, as usual your mileage may vary and you should examine the suitability of Linux for your task − something which we hope this document helps you to accomplish.

2. Getting this FAQ

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.

3. What hardware does Linux/MIPS support?

3.1 Hardware Platforms

Many machines are available with a number of different CPU options of which not all are currently supported. Please check section Processor Types to make sure your CPU type is supported. This is a listing of machines that are running Linux/MIPS, systems to which Linux/MIPS could be ported, or systems that people have an interest in running Linux/MIPS.

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/.

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. Basically, the Qube kernel was derived from Linux/MIPS 2.1.56, backported to 2.0.30 for stability's sake, then optimized. Cobalt kernels are available from Cobalt's ftp site http://www.cobaltnet.com. The Cobalt Qube support has never been integrated into the official Linux/MIPS 2.1.x kernels.

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.

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/.

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 not 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 Indy

The Indy is currently the only (mostly) supported Silicon Graphics machine. The only supported graphics card is the Newport card, a.k.a. ``XL'' graphics. The Indy is available with a large number of CPU options at various clock rates, all of which are supported. There is also a X server available, now written by Guido Guenther (guido.guenther@gmx.net). If you're able to use the Newport console on your Indy it should be possible to also use this X server. It's based on XFree86 4.0 and currently runs at snail−like speeds, but it seems to work quite well. If you want to try it, take a look at http://honk.physik.uni−konstanz.de/~agx/mipslinux/x/ .

Strange amounts of available memory

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.

ELF support in old PROM versions

Old PROM versions don't know about the ELF binary format which the Linux kernel uses, so Linux cannot boot directly. The preferable solution for this is of course a PROM upgrade. Alternatively, you can use Sash for IRIX 5 or newer 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 should 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. That means if you intend to use Sash for booting the kernel from local disk, you'll still have to have a minimal IRIX installation on your system.

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.

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. The code is available in the Linux/MIPS CVS tree.

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 ...

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.

Other Silicon Graphics machines

At this time, no other Silicon Graphics machine is supported. This also applies to the *very* old Motorola 68k−based systems.

Sony Playstation

The Sony Playstation is based on an R3000 derivative and uses a set of graphics chips developed by Sony themselves. While the machine, in theory, is capable of running Linux, a port is difficult since Sony so far hasn't provided the necessary technical information. This still leaves the question of whether the port would be worthwhile. So in short, nothing has happened yet even though many people have shown an interest in running Linux on this system.

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.

Algorithmics P−4032, P−5064, P−6032

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

- P−6032 is a new board for CPUs with 32−bit buses (QED RM5231, NEC Vr43x0, NEC Vr5432, IDT 64x74)
- P−4032 is an older board obsoleted by P−6032.
- P−5064 is for CPUs with 64−bit buses, notably QED's RM70xx and RM52xx series.

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.

At the time of writing (November 2000) we are using a 2.2.x kernel; kernel code is shared with the ports being done by people from MIPS Technologies, Inc.). 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' 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

You can contact us at Algorithmics.

DECstation series

During the late 80's and the early 90's, Digital (now Compaq) built MIPS−based Workstations named DECstation resp. DECsystem. Other x86 and Alpha−based machines were sold under the name DECstation, but these are obviously not the subject of this FAQ. Support for DECstations is still under development, started by Paul M. Antoine. These days, most of the work is done by Harald Koerfgen (Harald.Koerfgen@home.ivm.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.

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 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://ftp.fnet.fr.

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.

3.2 Processor types

R2000, R3000 family

The R2000 is the original MIPS processor. It's a 32 bit processor which was clocked at 8MHz back in '85 when the first MIPS processors came to the market. Later versions were clocked faster, for example, the R3000 is a 100% compatible redesign of the R2000 which is just clocked faster. Because of their high compatibility, where this document mentions the R3000, in most cases the same facts also apply to the R2000. The R3000A is basically an R2000, plus an R3010 FPU, and 64K cache running at up to 40MHz all integrated onto the same chip.

Harald Koerfgen (Harald.Koerfgen@home.ivm.de) and Gleb O. Raiko (raiko@niisi.msk.ru) have both independently worked on patches for R3000 processors. The work has been merged and integrated into the official Linux/MIPS sources since July 1999. Actually, Linux supports R3000 processors including some derivatives like the R3081 and the TMPR3912/PR31700

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; a pretty 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.

R4000 and R5000 family

Linux supports many of the members of the R4000 family. Currently, these are: R4000PC, R4400PC, R4300, R4600, R4700, R5000, R5230, R5260. Many others are probably supported as well.

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.

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.

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

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 litte 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.

Processors with partial or no FPU

In theory, these processors are supportable, especially when applications don't rely on the presence of a FPU. We, however, don't support that yet.

3.3 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.linuxppc.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, probably 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.

---

4. Linux distributions.

4.1 RedHat

For MIPSeb, there's Rough Cuts Linux, previously known as Hard Hat Linux, which is most of RedHat Linux 5.1 ported for MIPSeb. You can get this at ftp://oss.sgi.com/pub/linux/mips/redhat.

It is bundled along with M68k, UltraSparc, and PowerPC in a package called "Rough Cuts" pressed by RedHat, and available wherever RedHat products are sold. This is a very convenient way to get it without having to download 280MB. Redhat is no longer offering this product but if you're lucky you may still find the CD in some shops or somewhere on the net.

A distribution based on Red Hat 5.2 that's targeting the Cobalt Qubes. Those binaries will work perfectly on other MIPSel architectures and are available at ftp://intel.cleveland.lug.net/pub/Mipsel. ftp://bolug.uni−bonn.de/mips has various rpm packages from Redhat 6.0, 6.1 and 6.2.

4.2 Debian

A Debian port is underway. Current efforts are being bootstrapped using SGI/Linux as a base, and dpkg compiles natively with few changes. In addition to the SGI version, some interest has been shown in little endian platforms. Keep an eye on the Debian−MIPS Port page, http://www.debian.org/ports/mips/ for developments.

4.3 Simple

This distribution is for big−endian systems only so far. It's highly experimental, intended for developers' use in testing the latest versions of gcc, binutils, glibc, and the kernel. This is the only glibc 2.2−based distribution available for MIPS. You can always get the latest version of this distribution and accompanying release notes at ftp://oss.sgi.com/pub/linux/mips/mips−linux/simple. Also available is a cross−compiler system to aid in development.

4.4 Other

Kevin Kissel of MIPS, Inc. has put together a compilation of the RedHat 5.2 mipsel distribution that includes the all the packages from the Cleveland LUG site ftp://intel.cleveland.lug.net/pub/Mipsel, plus quite a few others from elsewhere that are either newer than, or missing from, the Cleveland LUG distribution. This compilation can be found via HTML at http://www.paralogos.com/mipslinux or directly by FTP at ftp://ftp.paralogos.com/pub/linux/mips/RPMS/mipsel.

---

5. Linux/MIPS net resources.

5.1 Anonymous FTP servers.

The two primary anonymous FTP servers for Linux/MIPS are

oss.sgi.com

This server should satisfy almost all of your Linux/MIPS related ftp desires. Really.

ftp.fnet.fr

> This server is currently pretty outdated. It's included here mostly for completeness, and for people with interest in prehistoric software.

On all of these ftp servers, there is a list of mirror sites you may want to use for faster access.

Another source for little endian MIPS binaries is ftp://intel.cleveland.lug.net/pub/Mipsel, which carries mostly newer versions of binaries for the RedHat flavour shipping with the Cobalts.

5.2 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>.

The other important CVS archive of the Linux community is vger.kernel.org, where a lot of code is being collected before being sent to Linus for distribution. Although vger itself no longer offers anonymous access, there are mirror sites which do provide anonymous access. For details on how to access them, see http://cvs.on.openprojects.net/. The modules which are of interest are: ``linux'', ``modutils'', ``pciutils'', and ``netutils''.

5.3 Web servers.

The two primary web servers for Linux/MIPS are

http://oss.sgi.com/mips

> This server covers most of Linux/MIPS. It's somewhat SGI−centric, but since Linux/MIPS tries to be the same on every platform, most of its information is of interest to all users.

http://www.linux−mips.org

> Quite new site which one day will hopefully become the main Linux/MIPS site.

http://lena.fnet.fr

> This server is currently pretty outdated. It's included here mostly for completeness.

All of these servers have mirrors scattered all over the world − you may want to use one for best performance.

5.4 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.

5.5 Mailing lists.

There are three Linux/MIPS−oriented mailing lists:

linux−mips@fnet.fr

> This mailing list is used for most all non−SGI related communication. Subscription is handled by a human, and you can send your subscription requests to linux−mips−request@fnet.fr. You can unsubscribe from this mailing list by sending *unsubscribe <your−email−address>* to the same address. *Only subscribers are allowed to post to this list.*

linux−mips@oss.sgi.com

> This mailing list currently has the most traffic. It's somewhat SGI−centric but is nevertheless of interest especially to developers as a good number of SGI engineers are subscribed to this list. Subscription to this list is handled via Majordomo (majordomo@oss.sgi.com), 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/email.html.

5.6 IRC channel.

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

---

6. Installation of Linux/MIPS and common problems.

6.1 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.

6.2 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
```

6.3 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.

6.4 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).

6.5 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.

6.6 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.

6.7 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.

---

7. 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''.

Plans are to remove the need for Milo in the near future.

7.1 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.

7.2 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.

---

8. 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.

9. How do I set up a cross−compiler?

9.1 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.

9.2 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.

9.3 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 addional, 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.

9.4 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.

9.5 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.

9.6 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>.

9.7 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.

9.8 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.

9.9 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.

9.10 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.

9.11 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.

9.12 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.

9.13 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.

9.14 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++.

9.15 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.

9.16 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. Further reports that help to isolate the problematic configuration are welcome.

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.

9.17 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. Related Literature

10.1 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:

- http://www.algor.co.uk/algor/info/seemipsrun.html (europe)
- http://www.mkp.com/books_catalog/1−55860−410−3.asp (US)

and from good bookshops anywhere. It's 512 pages and costs around $50 in the US, £39.95 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:

10.2 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.

10.3 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.

10.4 UNIX System V ABI MIPS Processor Supplement

By Prentice Hall, Published 05/1991, ISBN 0−13880−170−3. This book defines many of the MIPS specific technical standards like calling conventions, ELF properties, and much more that is being used by Linux/MIPS. Unfortunately it's out of print. Similarly, the site *"http://www.mipsabi.org/"* is offline.

10.5 The mips.com site

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

10.6 The NEC site

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