# The Various Ports

This document collects comments about the various architectures supported by Plan 9. The system tries to hide most of the differences between machines, so the machines as seen by a Plan 9 user look different from how they are perceived through commercial software. Also, because we are a small group, we couldn't do everything: exploit every optimization, support every model, drive every device. This document records what we *have* done. The first section discusses the compiler/assembler/loader suite for each machine. The second talks about the operating system implemented on each of the various machines.

### The MIPS compiler

This compiler generates code for the R2000, R3000, and R4000 machines configured to be big-endians. The compiler generates no R4000-specific instructions although the assembler and loader support the new user-mode instructions. There are options to generate code for little-endian machines. Considering its speed, the Plan 9 compiler generates good code, but the commercial MIPS compiler with all the stops pulled out consistently beats it by 20% or so, sometimes more. Since ours compiles about 10 times faster and we spend most of our time compiling anyway, we are content with the tradeoff.

The compiler is solid: we've used it for several big projects and, of course, all our applications run under it. The behavior of floating-point programs is much like on the 68040: the operating system emulates where necessary to get past non-trapping underflow and overflow, but does not handle gradual underflow or denormalized numbers or not-a-numbers.

### The SPARC compiler

The SPARC compiler is also solid and fast, although we haven't used it for a few years, due to a lack of current hardware. We have seen it do much better than GCC with all the optimizations, but on average it is probably about the same.

We used to run some old SPARC machines with no multiply or divide instructions, so the compiler does not produce them by default. Instead it calls internal subroutines. A loader flag, −M, causes the instructions to be emitted. The operating system has trap code to emulate them if necessary, but the traps are slower than emulating them in user mode. In any modern lab, in which SPARCS have the instructions, it would be worth enabling the −M flag by default.

The floating point story is the same as on the MIPS.

### The Intel i386 compiler

This is really an *x*86 compiler, for $x>2$. It works only if the machine is in 32-bit protected mode. It is solid and generates tolerable code; it is our main compiler these days.

Floating point is well-behaved, but the compiler assumes i387-compatible hardware to execute the instructions. With 387 hardware, the system does the full IEEE 754 job, just like the MC68881. By default, the libraries don't use the 387 built-ins for transcendentals. If you want them, build the code in `/sys/src/libc/386/387`.

**The AMD64 compiler**

The AMD64 compiler has been used to build 64-bit variants of Plan 9. It seems to be reasonably solid.

**The PowerPC compiler**

The PowerPC compiler supports the 32-bit PowerPC architecture only; it does not support either the 64-bit extensions or the POWER compatibility instructions. It has been used for production operating system work on the 603, 603e, 604e, 821, 823, and 860, and experimental work on the 405, 440 and 450. On the 8xx floating-point instructions must be emulated. Instruction scheduling is not implemented; otherwise the code generated is similar to that for the other load-store architectures. The compiler makes little or no use of unusual PowerPC features such as the counter register, several condition code registers, and multiply-accumulate instructions, but they are sometimes used by assembly language routines in the libraries.

**The PowerPC64 compiler**

The PowerPC64 compiler supports the 64-bit PowerPC architecture only. It has been lightly used on IBM's Blue Gene machines.

**The ARM compiler**

The ARM compiler is fairly solid; it has been used for some production operating system work including Inferno and the Plan 9 kernel for the iPAQ, which uses a StrongArm SA1, and the Sheevaplug, Guruplug, Dreamplug, Gumstix Overo, Compulab Trimslice and others. The compiler supports the ARMv4 and later 32-bit architectures; it does not support the Thumb instruction sets. It has been used on ARM7500FE, ARM926 and Cortex-A8 and -A9 processors and the Strongarm SA1 core machines. The compiler generates instructions for ARM 7500 FPA floating-point coprocessor 1 by default, but `5l -f` instead generates VFP instructions for coprocessors 10 and 11.

**The IBM PC operating system**

The PC version of Plan 9 can boot via PXE or directly from a disk created by the `format` command; see *prep*(8). Plan 9 runs in 32-bit mode—which requires a 386 or later model x86 processor—and has an interrupt-driven I/O system, so it does not use the BIOS (except for a small portion of the boot program and floppy boot block). This helps performance but limits the set of I/O devices that it can support without special code.

Plan 9 supports the ISA, EISA, and PCI buses as well as PCMCIA and PC card devices. It is infeasible to list all the supported machines, because the PC-clone marketplace is too volatile and there is no guarantee that the machine you buy today will contain the same components as the one you bought yesterday. (For our lab, we buy components and assemble the machines ourselves in an attempt to lessen this effect.) IDE/ATA, SATA and SCSI disks are supported. CD-ROMs are supported two ways, either on the SCSI bus, or as ATA(PI) devices. The SCSI adapter must be a member of the Mylex Multimaster (old Buslogic BT-*) series or the Symbios 53C8XX series.

Supported Ethernet cards include the AMD79C790, 3COM Etherlink III and 3C589 series, Lucent Wavelan and compatibles, NE2000, WD8003, WD8013, Realtek 8139, SMC Elite and Elite Ultra, Linksys Combo EthernetCard and EtherFast 10/100, and a variety of controllers based on the Intel i8255[789] and Digital (now Intel) 21114x chips. We support Gigabit Ethernet via Realtek 8110S/8169S, and Intel 8254[013467], 8256[36], and 8257[1-79] controllers. We support 10-Gigabit Ethernet via Intel's 8259[89], and Myricom's 10g-pcie-8a. We mostly use Intel and Realtek gigabit controllers, so those drivers may be more robust.

There must be an explicit Plan 9 driver for peripherals; it cannot use DOS or Windows drivers. Plan 9 cannot exploit special hardware-related features that fall outside of the IBM PC model, such as power management, unless architecture-dependent code is added to the kernel. For more details see *plan9.ini*(8).

Over the years, Plan 9 has run on a number of VGA cards. Recent changes to the graphics system have not been tested on most of the older cards; some effort may be needed to get them working again. In our lab, most of our machines use the ATI or Nvidia chips, so such devices are probably the most reliable. The system requires a hardware cursor. For more details see *vgadb*(6) and *vga*(8). The wiki (`http://plan9.bell-labs.com/wiki/plan9`) contains the definitive list of cards that are known to work; see the ''supported PC hardware'' page.

For audio, Plan 9 supports the Sound Blaster 16 and compatibles. (Note that audio doesn't work under Plan 9 with 8-bit Sound Blasters.) There is also user-level support for USB audio devices; see *usb*(4).

Finally, it's important to have a three-button mouse with Plan 9. The system currently works only with mice on the PS/2 port or USB. Serial mouse support should return before long.

Once you have Plan 9 installed (see the wiki's installation document), use PXE or a boot disk to load the system. See *booting*(8), *9boot*(8), and *prep*(8) for more information.

### The Routerboard 450G operating system

This is a CPU kernel that runs on the Mikrotik Routerboard RB450G, which contains a MIPS 24K CPU (the Atheros 7161), which implements the MIPS32R2 architecture. It has 256MB of RAM and a serial port. The CPU lacks the 64-bit instructions of previous MIPS systems (e.g., SGI Challenge and Carrera). There is no hardware floating-point, so we emulate the instructions. Only the first of the five Gigabit Ethernet ports is currently supported; the other four are connected via an internal switch. To avoid a bug in the CPU (erratum 48), we run the caches write-through, rather than write-back, and compiled `/mips` with a *vl* modified to emit enough NOPs to avoid three consecutive store instructions (see `/sys/src/cmd/vl/noop.c` to enable this).

### The PowerPC operating system

We have a version of the system that runs on the PowerPC on a home-grown machine called Viaduct. The Viaduct minibrick is a small (12x9x3 cm) low-cost embedded computer consisting of a 50Mhz MPC850, 16MB sdram, 2MB flash, and two 10Mb Ethernet ports. It is designed for home/SOHO networking applications such as VPN, firewalls, NAT, etc.

The kernel has also been ported to the Motorola MTX embedded motherboard; that port is included in the distribution. The port only works with a 604e processor (the 603e is substantially different) and at present only a single CPU is permitted.

We have ports to the Xilinx Virtex 4 and 5 FPGAs which use PowerPC 405 and 440 processors, respectively.

### The Marvell Kirkwood operating system

This is an ARM kernel for the ARM926EJ-S processor and it emulates ARM 7500 floating-point and CAS (compare-and-swap) instructions. It is known to run on the Sheevaplug, Guruplug, Dreamplug and Openrd-client boards. It is derived from a port of native Inferno to the Sheevaplug by Salva Peiró and Mechiel Lukkien. There are many features of the Kirkwood system-on-a-chip that it does not exploit. There are currently drivers for up to two Gigabit Ethernet interfaces, USB and the console serial port; we hope to add crypto acceleration, and a video driver for the Openrd-client.

### The Marvell PXA168 operating system

This is an ARM kernel for the ARM–v5–architecture processor in the Marvell PXA168 system–on–a–chip and it emulates ARM 7500 floating–point and CAS (compare–and–swap) instructions. It is known to run on the Guruplug Display. There are many features of the system–on–a–chip that it does not exploit. There are currently drivers for a Fast Ethernet interface, and the console serial port; we hope to add crypto acceleration, and a video driver.

### The TI OMAP35 operating system

This is an ARM kernel for the Cortex–A8 processor and it emulates ARM 7500 floating–point and CAS (compare–and–swap) instructions. It is known to run on the IGEPv2 board and the Gumstix Overo, and might eventually run on the Beagleboard, once USB is working. There are many features of the OMAP system–on–a–chip that it does not exploit. Initially, there are drivers for the SMSC 9221 100Mb/s Ethernet interface in the IGEPv2 and Overo, and the console serial port; we hope to add USB, flash memory and video drivers.

### The Nvidia Tegra2 operating system

This is an ARM kernel for the dual Cortex–A9 processors in the Nvidia Tegra2 system–on–a–chip and it emulates ARM 7500 floating–point and CAS (compare–and–swap) instructions, but the hardware includes VFP3 floating–point. It runs on the Compulab Trimslice. There are many features of the system–on–a–chip that it does not exploit. Initially, there are drivers for the Ethernet interface and the console serial port; we hope to add USB, flash memory and video drivers.

### The Broadcom 2835 operating system

This consists of terminal and CPU kernels for the ARM1176 processor in the Broadcom 2835 system–on–a–chip. The hardware includes VFP2 floating–point. It runs on the Raspberry Pi Models A and B. Since it relies upon USB Ethernet and the Plan 9 USB Ethernet driver doesn't implement multicast, this port can't speak IPv6.