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 Motorola MC68020 compiler

This is the oldest compiler of the bunch. Relative to its competitors—commercial compilers for the same machine—it generates quite good code. It assumes at least a 68020 architecture: some of the addressing modes it generates are not on the 68000 or 68010.

We also use this compiler for the 68040. Except for a few instructions and registers available only from assembly language, the only user-visible difference between these machines is in floating point. Our 68020s all have 68881 or 68882 floating point units attached, so to execute floating point programs we depend on there being appropriate hardware. Unfortunately, the 68040 is not quite so thorough in its implementation of the IEEE 754 standard or in its provision of built-in instructions for the transcendental functions. The latter was easy to get around: we don't use them on the 68020 either, but we do have a library, `-l68881`, that you can use if you need the performance (which can be substantial: `astro` runs twice as fast). We don't use this library by default because we want to run the same binaries on both machines and don't want to emulate `FCOSH` in the operating system.

The problem with IEEE is nastier. We didn't really want to deal with gradual underflow and all that, especially since we had half a dozen machines we'd need to do it on, so on the 68040 we implement non-trapping underflow as truncation to zero and do nothing about denormalized numbers and not-a-numbers. This means the 68020 and the 68040 are not precisely compatible.

The Motorola MC68000 compiler

This compiler is a stripped-down version of the MC68020 compiler built for an abortive port to the Dragonball processor on the Palm Pilot. It generates position-independent code whose overall quality is much poorer than the code for the MC68020.

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 Intel i960 compiler

This compiler was built as a weekend hack to let us get the Cyclone boards running. It has only been used to run one program—the on-board code in the Cyclone—and is therefore likely to be buggy. There are a number of obvious optimizations to the code that have never been attempted. For example, the compiler does not support pipelining. The code runs in little-endian mode.

The DEC Alpha compiler

The Alpha compiler is based on a port done by David Hogan while studying at the Basser Department of Computer Science, University of Sydney. It has been used to build a running version of the operating system, but has not been stressed as much as some of the other compilers.

Although the Alpha is a 64-bit architecture, this compiler treats `int`s, `long`s and pointers as 32 bits. Access to the 64-bit operations is available through the `vlong` type, as with the other architectures.

The compiler assumes that the target CPU supports the optional byte and word memory operations (the ''BWX'' extension). If you have an old system, you can generate code without using the extension by passing the loader the `-x` option.

There are a number of optimizations that the Alpha Architecture Handbook recommends, but this compiler does not do. In particular, there is currently no support for the code alignment and code scheduling optimizations.

The compiler tries to conform to IEEE, but some Alpha CPUs do not implement all of the rounding and trapping modes in silicon. Fixing this problem requires some software emulation code in the kernel; to date, this has not been attempted.

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 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 and others. The compiler supports the ARMv4 architecture; it does not support the Thumb instruction sets. It has been used on ARM7500FE, ARM926 and Cortex-A8 processors and the Strongarm SA1 core machines. The compiler generates instructions for ARM 7500 FPA floating-point coprocessor 1, but probably should instead generate VFP 3+ instructions for coprocessors 10 and 11.

The AMD 29000 compiler

This compiler was used to port an operating system to an AMD 29240 processor. The project is long abandoned, but the compiler lives on.

The Carrera operating system

We used to have a number of MIPS R4400 PC-like devices called Carreras, with custom-built frame buffers, that we used as terminals. They're almost all decommissioned now, but we're including the source as a reference in case someone wants to get another MIPS-based system running.

The IBM PC operating system

The PC version of Plan 9 can boot either from MS-DOS 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.) Both IDE/ATA and SCSI disks are supported, and there is support for large ATA drives. 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, 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 mostly use Etherlink III, i8255[789], and 21114x, so those drivers may be more robust. There must be an explicit Plan 9 driver for peripherals; it cannot use DOS or Windows drivers. Also, 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 Mach64, S3 ViRGE, or S3 Savage chips, so such devices are probably the most reliable. We also use a few Matrox and TNT cards. 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) run the program `ld` from DOS or use a boot disk. See *booting*(8), *9load*(8), and *prep*(8) for more information.

The Alpha PC operating system

Plan 9 runs on the Alpha PC 164. The Alpha port has not been used as much as the others, and should be considered a preliminary release.

The port uses the OSF/1 flavor of PALcode, and should be booted from the SRM firmware (booting from ARC is not supported). Supported devices are a subset of the PC ones; currently this includes DECchip 2114x-based ethernet cards, S3 VGA cards, Sound Blaster 16-compatible audio, floppy drives, and ATA hard disks.

The system has to be booted via tftp. See *booting*(8) for details.

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.

The Compaq iPAQ operating system

Plan 9 was ported to Compaq's iPAQ Pocket PC, which uses the StrongArm SA1 processor. The model we have is a 3630; neighboring models also work. The kernel can drive a PCMCIA sleeve with a WaveLAN card, but no other PCMCIA devices have been ported yet.

The iPAQ runs `rio` with a small keyboard application that allows Palm-style handwriting input as well as typing with the stylus on a miniature keyboard.

Fco. J. Ballesteros (`nemo@plan9.escet.urjc.es`) added support for hibernation, but we haven't been able to get that to work again in the new kernel; the code is there, however, for volunteers to play with. See the file `/sys/src/9/bitsy/Booting101` for information about installing Plan 9 on the iPAQ.

The Marvell Kirkwood operating system

This is an ARM kernel for the ARM926EJ-S processor and it emulates 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 Peir 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 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 pre-VFPv3 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.