Introducing IRIX 6.0

System Software Marketing, Computer Systems Group

This note discusses changes made to IRIX for Power Challenge to accommodate 64 bit programs and the R8000 architecture. It is targeted at developers to aid in their migration planning.

Executive Summary

IRIX 6.0 is the first version of IRIX for R8000 Power Challenge and Power Onyx systems. This system will support today's 32 bit user code binaries without change, and will permit the creation of binaries which use the 64 bit features of the new MIPS R8000 Streaming Super-scalar RISC family.

IRIX 6.0 will support a C programming model in which the *long* and *long long* data types and all pointers are 64 bits, this strategy was chosen as the best overall choice. The programming model decision has ramifications in the definition of library interfaces, e.g. all pointers are 64 bits. The migration to 64 bit provides the opportunity to increase the limits on the process size, and system memory configuration. While support for 8 GB file systems and 2 GB files is provided at first release, support for sizes which exceed these limits will be provided in a subsequent release.

Application developers have unparalleled flexibility in their options under IRIX 6.0. Since 32 bit binaries are supported, the application developer is not forced to migrate to 64 bit environments. Applications which wish to take advantage of R8000 floating point performance, or 64 bit features such as 64 bit logical operations and very large virtual address spaces should consider migration.

If migration to the full 64 bit environment is planned, preparation can begin immediately by ensuring that modern programming practices are followed, such as strict use of ANSI C function prototypes and using system-supplied type definitions. Another key preparation step is the definition of file data structures and protocol streams which are independent of the word size. With such preparation, the migration can be completed in most cases by simply re-compiling in the new environment when IRIX 6.0 is available.

Introduction

The purpose of this paper is to provide a technical outline of the features available in IRIX 6.0. This includes information on the R8000 processor, the IRIX 6.0 Development Option (IDO), changes made to IRIX to support 64 bits, and advantages of migrating applications to this operating system and architecture. Compatibility with existing versions of IRIX, and the release schedule for SGI layered software products will also be discussed.

Design Goals

What does Silicon Graphics supply to support 64 bit?

There are three fundamental requirements for 64 bit support:

- a processor which supports 64 bit data types efficiently. This implies that all the primary data paths on the processor chip and external to it are (at least) 64 bits wide. This processor also needs to support 64 bit virtual address space.
- a compilation system which can produce 64 bit executables using 64 bit virtual addressing
- an operating system environment that can run 64 bit programs

The R8000 processor fully supports 64 bit data types and 64 bit virtual address space. The new MIPSpro compiler will have full support of all the 64 bit data types and 64 bit virtual addressing. Finally IRIX 6.0, SGI's multi-threaded SMP version of the UNIX operating system, will complete the 64 bit computing environment to allow fully 64 bit programs to be run on SGI platforms.

Introducing the R8000 Processor

The R8000 microprocessor chip set is the first 64-bit superscalar implementation of the MIPS 4 Instruction Set Architecture (ISA), which is a superset of the MIPS 3 architecture and is backward compatible. Its superscalar implementation achieves extremely affordable supercomputer performance. The R8000 processors combine a large fast-access, high-throughput cache subsystem with high-performance floating-point capabilities. Its bandwidth satisfies applications with large working sets of data. The R8000 system is able to sustain levels of performance once only possible on much larger and more costly supercomputer systems.

The key features of the R8000 chip set include:

- Four-way superscalar architecture, capable of executing six operations per clock cycle
- True 64-bit microprocessor with 64-bit integer and floating-point operations, registers, and virtual addresses

- On chip floating point instruction cache
- Multi-component chip set consisting of an integer unit (IU), floating point unit (FPU), tag RAMS, and on Power Challenge 4MB of data streaming cache
- 16KB of instruction cache (I-cache) in IU, 16KB of dual-ported data cache (D-cache) in IU, 1K entries of branch prediction cache
- Memory Management Unit (MMU) in IU contains a 384-entry, dual-ported, three-way set associative Translation Lookaside Buffer (TLB)
- ANSI/IEEE-745 standard floating-point coprocessor with imprecise interrupts
- Full compatibility with earlier 32-bit and 64-bit MIPS microprocessors

TABLE 1. Advanced MIPS Instruction Sets

Set	Processor	Feature
MIPS 1	R3000	MIPS 1 Instruction Set
MIPS 2	R4000	MIPS 1 Instruction Set plus ldc1/sdc1, single instruction to load/store double precision branch likely
MIPS 3	R4400	MIPS 2 Instruction Set plus 64 bit integer operations 32 double precision floating point registers
MIPS 4	R8000	MIPS 3 Instruction Set plus MADD (Multiply and Add-Double) Indexed Floating Point load/store Conditional Move more Condition Code Registers

For more information see R8000 MicroProcessor Chip Set, Product Information

IRIX 6.0 Development Option (IDO)

IRIX 6.0 Development Option (IDO) includes new MIPSpro compiler technology to generate MIPS 3 or MIPS 4 64 bit code. The development environment integrates the well-known Silicon Graphics 32 bit compilers with new MIPSpro compiler technology to provide MIPS3 and MIPS 4 64 bit compilation. *Include* structures have been modified to provide the correct 32 or 64 bit sizes, and a hierarcy of libraries has been included to support both 32 bit and 64 bit development. This structure is described later in this paper.

Extensions include the definition of a new Extended Linking Format (ELF) object module format for 64 bit programs, ELF-64, which is supported in all the tools which work with object modules. ELF-64 is a 64 bit version of the ELF format used in IRIX 5.2 and other System V Release 4 systems. The Dynamic Shared Object (DSO) architecture was also extended to support 64 bits. DSO technology is utilized to provide upward compatibility between IRIX 5 and IRIX 6, and provides the flexibility to have a single, "shrink wrapped" version that can operate without change on a wide variety of hardware platforms based on the MIPS processor family.

IRIX 6.0 includes the latest releases of the X11 Window System™, X11 Release 6 (X11R6) libraries support, and of the IRIX IM, Silicon Graphics enhanced version of OSF/Motif Release 1.2.4. These libraries incorporate changes made to support 64 bits and some bug fixes. The X server is X11R5 to provide compatible behavior with 32 bit X11R5 executables without recompilation. OpenGL is fully supported for 32 bit and 64 bit development.

64 bit type changes - The LP64 Model

A 64 bit operating system implies that pointers are 64 bits so the virtual address space of a process can (potentially) be 2^{64} instead of 2^{32} as in current systems. To eliminate ambiguity in what other types are 64 bits, Table 2 summarizes the new 64 bit and the current 32 bit environment. The current IRIX environment (and other 32 bit UNIX® environments) support the sizes shown in the first column. The second column indicates exactly what data types are 64 bits and what stay 32 bits. The model used for IRIX 6.0 is called the LP64 model[1], because pointers and longs are 64 bits. For reference, the environment for INTEL x86 family members is included.

With the C compiler, the code generated using the flags -64 -mips3 or -64 -mips4, for *long*/*void** or *longlong* now uses the processors 64bit integer instructions. The Fortran compiler has no data size changes, as its types have specific bit sizes, but -64 -mips3 or -64 -mips4 now generates the 64bit integer instructions and 64bit addresses. 32 bit code (MIPS-1 or MIPS-2) will contain different size structures than 64 bit code (MIPS3 or MIPS4). The kernel handles this by converting all system calls from a 32 bit binary into 64 bit internal kernel format.

TABLE 2. C Data Type Models

C Data Type	current IRIX 32 bit	IRIX 6 LP64	Intel x86
char	8	8	8
short	16	16	16
int	32	32	16
long int	32	**64**	32
long long int	64	64	N/A
pointer	32	**64**	16/32

1. OSF/1 on DEC Alpha also supports the LP64 model.

C Data Type	current IRIX 32 bit	IRIX 6 LP64	Intel x86
float	32	32	32
double	64	64	N/A
long double[a]	64	**128**	N/A

a. On 32 bit systems use of the *long double* data type generates a warning message, that the long qualifier is not supported.

Available 64 bit SGI languages and CASE Tools

Languages supported at first release of IRIX 6.0 are: Assembler, C, C++, and Fortran77 (Fortran 90 in beta). Note the C++ is a native compiler, not a preprocessor.

Silicon Graphics CASEVision/WorkShop debugger will support debugging 64 bit applications, and there is a new ProMPF performance analysis tool for parallized Fortran to support 64 bits on IRIX 6.0. Both the Static Analyzer and the Build Analyzer will run on IRIX 6.0, however the Performance Analyzer and Coverage Analyzer will not be available at first release.

New Procedure Calling Convention

The MIPSpro compilers have introduced other new efficiencies in emitted code which may benefit applications. These are:

- 8 integer argument registers are available
- 32 floating point registers
- global pointer (*gp*) is saved by the callee
- small structures are passed in registers
- floating point arguments are more efficiently passed

New 64 bit Fortran library support

The Challenge complib set of libraries includes:

- Basic Linear Algebra Subprograms (BLAS) Libraries
- LAPACK, LINPACK, EISPACK Libraries
- Signal Processing Libraries for Fast Fourier and Convolutions
- SCIPORT (Portable version of SCILIB from Cray Research)
- SLATEC Mathematical Subprogram Library from the Energy, Science and Technology Software Center

64 Bit X11R6 and IRIX IM

To support Graphics developers, 64 bit versions of X11R6 libraries, and IRIX IM (enhanced OSF/Motif 1.2.4). Developer utilities have been modified for 32 and 64 bit development, including new imake config files, *xmkmf*, and the *uil* user interface language compiler. The 64 bit version of libXm implements standard Motif style, that is SGI style guide extensions are not included. Consequently some users may notice slight visual differences between 32 and 64 bit Motif applications.

64 bit OpenGL

OpenGL is a graphics software library that allows programmers to produce high quality color images of 3D objects. The OpenGL standard is governed by the Architecture Review Board, an industry consortium responsible for defining OpenGL.

IRIX 6.0 includes a 64 bit version of OpenGL include files and libraries to facilitate the development of 64 bit OpenGL applications. The 32 bit version of the IRIS GL and OpenGL libraries are supplied so that customers can execute their 32 bit applications. As IRIS GL is superceeded by OpenGL, 64 bit version of IRIS GL libraries are not supplied. Software with GL that need 64 bits will need to port to OpenGL The GL to OpenGL translation tool *toogl* is provided, and additional information is available in the <u>OpenGL Optimization and Porting Guide</u>.

64 bit IRIX 6.0 Features

Large (>2GB) Processes

A MIPS3 or MIPS4 user process can grow to a virtual address space of 2^{40} Bytes (1 TB), provided the system has sufficient memory and swap space. This permits very large tasks to be run without segmentation.

Large (>2GB) Physical and Virtual Memory

With IRIX 6.0 physical memory can be up to 16 GBytes (the current Challenge system limit). This permits more tasks to be run simultaneously than is possible on a 32 bit operating system.

Support for 64 bit integer and logical operations

User programs with 64 bit operations (potentially) double in performance when compared to executing them in a 32 bit environment. This improvement is due to larger bitwise logical operations and the ability to fit larger numbers into a single integer operation.

Mature Multiprocessor Support for the R8000 Processor

IRIX 6.0 will support up to 18 R8000 processors per Challenge/Onyx system, providing unparalleled computing performance especially on floating point intensive programs. The IRIX 6.0 kernel is optimized for the R8000 instruction set, and 64 bit operations.

One optimization for the R8000 was to adopt a 16KB kernel page size. This matches the R8000 cache architecture, and results in fewer pages, and better SPEC performance.

As MP versions of IRIX have been available since 1987, IRIX 6.0 MP support is based on significantly more mature technology than other systems vendors. Within the IRIX kernel there are over 4000 distinct MP safe entry points, allowing applications to achieve near linear performance improvements with additional processors. This kernel technology is supported by advanced parallelizing compilers, and MP CASE tools, which enable developers to take advantage of these features.

Compatibility between IRIX 5.2 and IRIX 6.0

IRIX 6.0 was created within the framework of IRIX 5.2. Features and changes relevant to 64 bit operation were added, as well as a few post 5.2 bug fixes. Hence its 32 bit support could be considered as 5.2+. Nearly all IRIX 4 and IRIX 5 (including MIPS ABI) binaries can run under IRIX 6. There may be rare cases in which application code must be recompiled, or even less likely, recoded to take advantage of R8000 enhancements.

The compiler tools in IRIX 6.0 generate MIPS1 or MIPS2 32 bit code which can be used on IRIX 6.0 (but are not guaranteed to be backwards portable to a 5.2 system). MIPS1 or MIPS2 code generated on a 5.2 system can be used on IRIX 6.0. In general, development cannot be done on IRIX 6.0, of programs to run on 5.2.

It is also not possible to link objects compiled for 32 bits with objects or libraries compiled for 64 bits within a single program, because of the differences in the size of the calling arguments.

Device drivers and other applications using kernel structures

Applications with knowledge of kernel structure sizes, should port to 64 bits for two reasons. The first is that the size and layout of these structures may have changed, and second is the pointer to the structure is now a 64 bit quantity. Device drivers should be ported to 64 bits because the kernel is itself a 64 bit object. System structures changed to support 64 bits are listed in the section “Include Files and Library Directory Structures”.

Programs with Floating Point Exception Handlers

The R8000 microprocessor architecture defines two execution environments which effect applications which utilize their own exception handlers. In Performance Mode, the processor enhances the speed of the application by trapping exceptions imprecisely, this will not effect applications which do not cause floating point exceptions. Precise Exception Mode, is fully compatible with the existing MIPS floating point architecture, and is enabled by using the *fpmode* command.

Real Time Performance

IRIX 5.2 REACT™ features are provided in IRIX 6.0 and functionally complete, however there is no guaranteed interrupt latency performance. Later releases of IRIX 6.0 will include enhancements as required to meet IRIX 5.2 interrupt latency specifications.

Page Size

For R8000 systems, the system page size in IRIX 6.0 was increased from 4KB to 16KB to improve transfer speeds for large applications. Unfortunately some applications hard code this value as 4K rather than using the getpagesize() system call which will return the correct value.

New API for 32 bit programs to access 64 bit files:

Support for greater than 32 bit file systems would be expected as a generic 64 bit feature. Silicon Graphics' Extent File System (EFS) limits file systems to 8GB, and files to 2GB. In preparation for a 64 bit file system, interfaces have been added to allow both 32 bit and 64 bit applications access to 64 bit file system features.

In order to support the access of 64 bit files from 32 bit applications, new interfaces must be defined which take 64 bit parameters. These interfaces are cleanly supported in the kernel, without the information of whether an application is 64 or 32 bit filtering down below the system call level.

It is important to understand that the need for these extensions is only to allow 32 bit applications to access 64 bit files. They are not needed by 64 bit applications or by 32 bit applications which have no need to deal with 64 bit files.

TABLE 3. New 64 Bit User Types

off64_t	**Used for specifying 64 bit file offsets and sizes**
stat64	Returns a stat64 structure with 64 bit offset
flock64	Allows record locking on large files
rlim64_t, rlimit64	Supports setrlimit(2) and getrlimit(2) with 64 bit values
fpos64_t	Stdio fsetpos() and fgetpos() 64 bit offset support

TABLE 4. New and Extended System Calls and Library Interfaces

stat64(),fstat64() lstat64()	argument is a pointer to a stat64 structure
truncate64(),lseek64() ftruncate64(),mmap64()	uses off64_t to specify the new length of the file
getrlimit64, setrlimit64()	use the new rlimit64 structure
fcntl()	takes pointer to a flock64 structure

TABLE 4. New and Extended System Calls and Library Interfaces

stat64(),fstat64() lstat64()	argument is a pointer to a stat64 structure
fseek64(), ftell64()	uses 64 bit value for file offset
fgetpos64(),fsetpos64()	uses pointers to fpos64_t parameters

32 bit programs either recognize the existence of large files or they do not. Those that do will be using new interfaces to access these files. Those that do not will see all files as having a maximum size of 2GB-1 bytes. Data beyond offset 2GB-1 in a file will be inaccessible by applications which do not recognize the existence of large files, and the size of a file which is greater than 2GB-1 will not be visible to an application which does not know of such files.

With respect to NFS. The offset is explicit in each NFS request and is a 32 bit quantity. Sun has proposed 64 bit extensions to NFS as part of the ONC+ package. SGI has stated that our strategic direction is to implement ONC+ features as soon as they are made available.

Advantages converting to 64 Bit

The first, and probably most important, message is that you are not required to convert applications to 64 bit. Unlike some other 64 bit implementations, the 64 bit version of IRIX will fully support today's 32 bit binaries. These current binaries can run without change on both R4000 family and R3000 family processors. These binaries can also run on Streaming Super-scalar RISC systems without change. Only if the application can benefit from the features and performance available to 64 bit programs, should it be converted.

- *Happy with 32 bit?*

Do nothing, 32 bit works fine.
Have a nice day.

- *Need 64 bit features?*

Start today to clean up the code to modern practices. Recompile on IRIX 6.0 and you're done.

- *Coming from a supercomputer?*

Wait for IRIX 6.0 and use compatibility support for Cray and Convex.

Basic Conversion Considerations

In making a choice of whether to convert to 64 bits or not, the following should be considered:

- can the application benefit from the R8000 floating point performance?
- can the application benefit from 64 bit logical and/or arithmetic operations or quad precision support?
- can the application benefit from a virtual address space larger than 32 bits?
- can the application benefit from files larger than 32 bits?
- is the application being ported from a 64 bit (supercomputer) environment?

Obviously this is not an exhaustive list, but it touches on some of the strongest reasons to consider the possibility of 64 bit migration.

Advantages of the R8000 Processor

The 75MHz R8000 out performs the 150MHz R4400 two to four times in double precision floating-point operations. We have measured performance in “real” floating-point intensive applications such as Chemistry, GIS, CAD/CAM, CFD, and FEA. Please refer to the Power Challenge performance brief for more information.

64 bit Arithmetic and Logical Operation Speedups

The ability to do logical operations in 64 bit strides can potentially double the speed of critical inner loops in applications such as image processing or ECAD circuit simulations where bits in vectors or matrices are logically manipulated. To exploit this capability, generally very minor surgery to the program is required. Simple changes to the declarations of the data structures and alteration of loop bounds will usually suffice.

The efficiency of supporting 64 bit integer arithmetic is also potentially more than doubled in 64 bit operation (as compared to doing the same operation in a 32 bit environment). As with the logical operations, some change to the software is required to exploit this capability, but it will be typically quite localized.

Large Address Spaces

Applications are beginning to require more than 32 bits of addressability to achieve their required performance or functionality. For example, a data base system may wish to employ a very large page cache to improve system performance. In some of today’s CAE environments, the 32 bit limitation constrains the size of the system being modeled. Removing this limitation can permit larger, more complex systems to be modeled. Application developers should be aware that there are secondary costs associated with the decision to migrate to 64 bits. Notably, the storage for pointers is doubled, even if no other 64 bit features are used. This can affect finely tuned applications by increasing the size of certain structures so they no longer “fit” in, say, a page frame. These disadvantages may well be offset by some of the secondary benefits of conversion to 64 bits. The most important of these are:

- improved register utilization, especially from the additional floating point registers available in 64 bit mode
- improved subroutine argument passing
- improved memory to memory move speed

Large Files and File Systems

Some applications, such as geophysics and particle physics, may require access to files which have a natural size greater than 32 bits (2 GBytes, since the offset is signed). One case where this is common is when the application must interact with supercomputer systems, which permit such large files. IRIX 6.0 will support files of up to 1 TByte when the new 64 bit file system is released.

Graphics Performance

If the user's graphics program performs a substantial number of floating point calculations then the R8000 processor may increase apparent graphics performance, however the underlying graphics hardware is the same for Onyx and Power Onyx. Any other performance gain might be due to secondary effects such as, larger cache size, or new efficiencies in the compiler optimizer.The same is true for X and IRIX IM. There are minor performance improvements in the server speed (which will be inherited in IRIX 5.3), but these do not require porting or recompilation. The 64 bit graphics support was provided primarily for customer with floating point code that is not separable from the graphics code. Otherwise the graphics code could be run as a 32 bit executable.

Super Computer Migration

This file size issue is one special case of the general issue of porting applications from other 64 bit supercomputer environments. Such applications are already adapted to 64 bits so are likely to need only minor change. (Indeed, reverting to 32 bits as is required by vendors whose systems are not 64 bit capable, such as Sun, HP and IBM, can be even more challenging than migrating to 64 bits from a 32 bit environment.) To aid in the migration, the compilation system will have optional features to provide source code compatibility, compatible compiler options, and compatible libraries with the Cray and Convex environments.

The compatibility libraries included with Fortran 6.0 contain many routines optimized for the MIPS architecture and hand-tuned to exploit maximum amount of parallelism available in the Power-Challenge system. These libraries are well documented and include basic linear algebra libraries, libraries for solving for dense linear systems, signal processing libraries, and a port of SCIPORT which includes routines commonly required for applications ported from the CRAY environment.

Migration from OSF/1

While OSF/1 is a 64 bit operating system, there are some differences which should be considered when migrating the application to IRIX 6.0. The intrinsic differences are the DEC version of OSF/1 is “little endian” and it also can be configured to permit nullpointers to be de-referenced in C routines. Correct type checking within the application will avoid problems in this area.

64 Bit Programming Considerations

Much of the preparation for 64 bit migration is simply employing good programming practices, such as the use of ANSI C function prototype declarations, and type definitions for the declaration of storage. Properly employed, these techniques permit a single source to be compiled for either 32 or 64 bit operation. Program style checkers such as lint can also be employed to help find potential trouble spots. Compiling an application with the *-fullwarn* option helps identify potential problems in 64 bit migration, such as questionable casts. This tool has been successfully used in the conversion of the IRIX kernel to the base for IRIX 6.0.

The following sections identify some broad issues which relate to porting an application to 64 bits. For more information see the SGI publication: Application Porting and Transition Guide to Power Challenge.

Don't Assume Data Type Sizes

The basic requirement is not to make assumptions about the absolute or relative size of *int*, *long* and pointer data types. It is sometimes the case that in older programs these assumptions are subtly built into the code, and the code will require modification to operate correctly in a 64 bit environment. In a 64 bit environment, the sizes of these data types differ, so they cannot be as freely interchanged as they can be in a 32 bit environment.

Probably the most commonly occurring problem is the assumption that *sizeof(int) == sizeof (void *)*. This can occur explicitly as in the case of a cast of a pointer to an *int*, or implicitly as a pointer is passed as an actual argument to a procedure where the formal argument is declared to be an *int*. It is also a common (bad) practice to use an *int* as an implicit union in structure declarations where either an *int* or a pointer may occur. In all of these cases, truncation of the high order 32 bits of a 64 bit address may occur. Careful attention to the warnings produced by the compiler about casts and rigorous use of ANSI C function prototypes can eliminate the problems between actual parameters and the formal declaration. In existing code, be especially careful about casts which have been inserted to "shut up" the warnings. These warnings are a nuisance in 32 bit environments, but in 64 bit environments, they represent impending disaster.

Pointer Arithmetic

A more subtle problem occurs when two pointers are subtracted to produce an offset. This will be a 64 bit quantity in a 64 bit environment, not an *int*. The proper declaration for such a result is the type definition *ptrdiff_t* from stddef.h. This type definition permits the program to work in either a 64 bit or 32 bit environment without change to the source.

Improper Union Use

Careful attention must be paid to union declarations which are used to access the structure of one of the basic types, e.g. the union of a four character array with a long. This has always been a bad practice due to endian issues, but it is much worse in a 64 bit environment. This practice is sometimes found when externally generated data such as protocol streams must be mapped into local constants. For example, a protocol header might be declared to be a union of a character array and a structure which contains the fields of the header. Although this style would have been non-portable from byte ordering (endian) issues, it may be used on code not intended to be ported to other architectures. Such a style will encounter problems in a 64 bit environment. Generally, shifts and masks must be used to extract fields from such streams.

Extensible Constants

The declaration of constants with the high order bit set may not produce the expected results if they are 64 bit quantities. For example 0xFFFFFFFF is just a large 64 bit integer, not -1 in a 64 bit environment. The recommended practice has long been to use extensible constants, but many programmers have not always followed the recommendation. This same type of problem occurs with declarations of bit masks for logical operation.

Converting FORTAN Applications to 64 bit

Converting a FORTRAN application to 64 bits is generally simpler than a C application. This is because there is less address manipulation in FORTRAN, and the size of data variables is well defined. In fact all the C rules mentioned above are applicable to FORTRAN, but there is less opportunity to incorrectly manipulate data of differing size. The default data types remain the same as in the 32 bit model, i.e. REAL and INTEGER are 32 bits. All data types ending *4, or *8 maintain their size. There are compiler flags for forcing the default size of INTEGER or REAL to 64 bits with the -i8 and -r8 flags respectively.

Applications which use the DEC extensions %LOC or %VAL need to be aware that the size of the addresses passed are 64 bit. Generally this is used with mixed language applications, for example FORTRAN and C. The developer should check that arguments passed between the routines are of equivalent size when mixing languages.

Differences between 32 bit and 64 bit Fortran Compiler passes

This section lists the differences between 32 bit and 64 bit Fortran compiler front ends. The 32 bit front end is called fcom, and the 64 bit front end is called fef77:

- REAL*4 constants are often promoted internally to REAL*8 in fcom, but not in fef77.
- Fcom allows more constant expressions in parameter statements.
- Fcom allows lines longer than 256 characters by default, fef77 is currently limited to 256 characters.
- Fef77 allows parameters (which are ignored) to the program statement.

- Fef77 allows empty parameters in subroutine/function calls which result in zeroes being passed by value.
- Fef77 implements Fortran 8x extensions to array syntax
- Fef77 implements REAL*16 and COMPLEX*32 and all associated intrinsics as 16 byte floating point entities. Fcom had recognized them, but converted them to REAL*8 and COMPLEX*16 respectively.
- Fef77 accepts PCF style parallel directives in addition to the directives such as C$DOACROSS which fcom accepts.

Hard to Find Problems

Some bad practices cannot be found easily by automated tools. For example, it is (unfortunately) not uncommon to see memory being allocated with an assumption about the size of the primitive object, e.g. malloc(4*count). Were the object in question one which was changing to 64 bits (8 bytes), the assumption would no longer be valid, and the program would be in error. Although the sophisticated CASE environments provided by CASEvision/WorkShop™ can be extremely helpful in finding these types of errors, they cannot substitute for human intervention and insight.

Run Time Issues

All of the above discussion has been on static issues. There are also potential pitfalls in the dynamics of the program operation. Migration to 64 bit may expose latent errors in the program such as overflows in arithmetic and shifts. These may not occur in the 64 bit environment, and if they were intended, appropriate mask operations must be performed. Similar problems can occur with sign or zero extension.

Machine Independent Data Files and Protocols

One final issue is the need to plan for data files and external protocol exchanges which are insensitive to the 32/64 bit issue. The systems will have ample processor power with which to support machine independent representations and to convert to the appropriate internal form as need be.

New MIPSpro Compiler Options

Language Independent new flags:

-32: compile for 32-bit ABI

-64: compile for 64-bit ABI

-A: add following cpp assertions

-H: run cpp and print list of headers

-W: -W<phase>,<arg> sends arg to phase {p=cpp,f=fe,b=be,a=as,l=ld}

-Y: -Y<phase>,<path> uses path to find phase
{p=cpp,f=fe,b=be,a=as,l=ld,I=include,S=startup,L=libraries}

-abi: generate strict-abi-compliant object files

-fe: stop after the front end is run

-help: print list of possible options

-help:: print list of possible options that contain given string

-keep: keep intermediate files

-mips3: compile for mips3 instruction set

-mips4: compile for mips4 instruction set

-mp: enable the multiprocessing directives

-nostdinc: no predefined include search path list

-nostdlib: no predefined library search path list

-show0: show what phases would be called, but don't invoke anything

-show1: show phases but don't have phases print additional info

-showt: show time taken by each phase

-woffall: turn off all warnings

-woffoptions: turn off warnings about options

-xgot: compile for extended GOT

New MIPSpro C Flags:

-Xa: extended ansi C

-Xc: strict ansi C

-Xt: K&R compatible C

New MIPSpro FORTRAN 77 Flags:

-d16: use double=8 and dcomplex=16

-d8: use double=8 and dcomplex=16

-noexpopt: Don't optimize exponentiation operations

-noi4: make integer and logical quantities short (2 bytes)

-pfakeep: run pfa and keep listing and mp file

-pfalist: run pfa and keep listing file

-r4: use REAL*4 and COMPLEX*8 as the defaults for real and complex

-r8: use REAL*8 and COMPLEX*16 as the defaults for real and complex

Flags that are not used in the new MIPSpro compiler.

A warning message is returned by the compiler for flags which are no longer necessary.

Include Files and Library Directory Structures

Since both 32-bit and 64-bit environments are supported in IRIX 6.0, there are multiple sets of libraries that reside in different directories. The system configuration for compiler defaults and library types will be optimized for speed. Therefore, 64-bit compilations will be the default. The defaults libraries will be MIPS4 for the R8000. Changing the library path will be necessary for other choices.

For all the header files under /usr/include, each file will support both 32-bit and 64-bit compilations. Products which cannot combine header files will have to come up with alternative naming solutions such as /usr/include/foo1/*.h and /usr/include/foo2/*.h.

The following directory structure will be used for IRIX 6.0 libraries:

TABLE 5. Directory Structures used in IRIX 6.0

Directories	Contents
/usr/lib/*.so	IRIX 5.2 DSOs
/usr/lib/*.a	IRIX 5.2 static libraries
/usr/lib/nonshared/*.a	IRIX 5.2 non-shared libraries
/usr/lib/mips2/*.so	IRIX 5.2 MIPS2 DSOs
/usr/lib/mips2/*.a	IRIX 5.2 MIPS2 static libraries
/usr/lib/mips2/nonshared/*.a	IRIX 5.2 MIPS2 non-shared libraries
/usr/lib64	64 bit libraries and tools
/usr/lib64/*.so	64 bit DSOs (symbolic link)
/usr/lib64/*.a	64 bit static libraries (symbolic link)
/usr/lib64/mips3/*.so	64 bit MIPS3 DSOs
/usr/lib64/mips3/*.a	64 bit MIPS3 static libraries
/usr/lib64/mips4/*.so	64 bit MIPS4 DSOs
/usr/lib64/mips4/*.a	64 bit MIPS4 static libraries
/usr/lib64/nonshared	64 bit non-shared libraries
/usr/lib64/mips3/nonshared/*.a	64 bit non-shared MIPS3 static libraries
/usr/lib64/mips4/nonshared/*.a	64 bit non-shared MIPS4 static libraries
/usr/lib64/cmplrs/*	64 bit compiler tools

TABLE 6. IRIX 6.0 System Structures extended to support 64 bits

<dirent.h>	<fcntl.h>	<ftw.h>	<grp.h>
<locale.h>	<math.h>	<netconfig.h>	<netdir.h>
<nl_types.h>	<poll.h>	<pwd.h>	<rpc.h>
<search.h>	<setjmp.h>	<sigaction.h>	<signal.h>
<stddef.h>	<stdio.h>	<stdlib.h>	<stropts.h>
<sys/ipc.h>	<sys/msg.h>	<sys/procset.h>	<sys/resource.h>
<sys/sem.h>	<sys/shm.h>	<sys/siginfo.h>	<sys/stat.h>
<sys/statvfs.h>	<sys/time.h>	<sys/times.h>	<sys/tiuser.h>

TABLE 6. IRIX 6.0 System Structures extended to support 64 bits

<sys/types.h>	<sys/uio.h>	<sys/utime.h>	<sys/utsname.h>
<termios.h>	<ucontext.h>		

SGI Software Availability for IRIX 6.0

The following is a list of software packages that will be provided in IRIX 6.0. Packages are provided in 64-bit versions to enable 64 bit development. The 32 bit versions are provided for execution of 32 bit applications built with IRIX 5, and 32 bit development under IRIX 6.0.

TABLE 7. IRIX 6.0 Software Packages

Packages	64 bit	32 bit
ADA	March'95	*
C++	*	*
CASEVision/Clearcase		
CASEVision/Tracker	compatible	*
CASEVision/WorkShop	*(most)	*
CASEVision/Workshop ProMPF	*	*
Complib	*	*
Compiler Libraries	*	*
Digital Media Development		*
Display PostScript Development		*
Documenters Workbench	compatible	*
Fortran	*	*
ImageVision Library		*
Impressario Development		*
Inventor	*	*
IRIS GL		*
Motif	*	*
NetLS Development		*
OpenGL	*	*
Pascal		*
Performer	*	*
PEX		*
Plot Libraries		*
Power C	*	*
Power Fortran	*	*

TABLE 7. IRIX 6.0 Software Packages

Packages	64 bit	32 bit
QuickTime Compression Libraries		*
Standard UNIX Libraries	*	*
SVR4 Networking Libraries		*
ToolTalk Development		*
X11 Libraries	*	*
Xau Libraries	*	*
Xext Libraries	*	*
Xmu Libraries	*	*
Xt Libraries	*	*

Product Positioning

IRIX 6.0 is the first release of IRIX which can fully exploit the advantages of the MIPS R8000 architecture. This release is intended to enable developers to compile floating point intensive programs optimized for the R8000, usually in Fortran, and be able to link to libraries they depend upon.

This IRIX release includes a new suite of compiler tools, which provide a merged development environment for 64 bit programs using the new MTI compilers, and 32 bit programs using the well known SGI compilers. The new MIPSpro compilers can generate 64 bit MIPS4 floating point optimized code, which can only execute on R8000 processors with IRIX 6.0.

Silicon Graphics will supply optimized versions of our own floating point intensive software, to show R8000 at its best. Note that a utility program has no need to be ported to 64bit, unless it needs to show floating point performance. However 64 bit applications can only link with 64 bit libraries.

At first release, in August 1994, there may be some useful libraries not available for 64 bit R8000 linking. Any omissions will be included in a subsequent bug fix release.

The purpose for maintaining the SGI compilers for 32 bit work is to maintain common 32 bit technology across all systems. To ensure that IRIX 6.0 is 32 bit compatible with other IRIX releases, the utility programs and libraries which are 32 bit are common to both releases. The first release of IRIX 6.0 will be closely tied to IRIX 5.2 for this reason. Silicon Graphics has no intention of dropping support for 32 bit development.

Silicon Graphics intends to converge the changes made to IRIX to support 64 bits on the R8000 with the version of IRIX for other MIPS architectures. Eventually, this will allow for a single operating system supporting a common 32 bit mode, and 64 bit capability on appropriate systems. There are no plans to provide 64 bit capability on older systems (other than support for the new 64 bit file system) which utilize the R3000 processor.

Bibliography

Application Porting and Transition Guide to Power Challenge Systems, drafts available from SGI.

R8000 MicroProcessor Chip Set, Product information available from MIPS Technologies Inc.

System V Applications Binary Interface, Revised Edition, UNIX Press (Prentice Hall), 1990, ISBN 0-13-880410-9. Defines the Generic ABI, mostly by reference to SVID 3.

System V Application Binary Interface, MIPS Processor Supplement, UNIX Press (Prentice Hall) 1990, ISBN 0-13-880170-3. Defines the MIPS processor implementation of the Generic ABI. A revision is planned. The changes are largely clarifications which come as a result of the MIPS ABI Group experience in making independently implemented systems binary compatible.

System V Interface Definition, Third Edition, AT&T/Prentice Hall 1989. (Five Volumes). Commonly referred to as SVID3, this set of documents define the SVR4 API.

MIPS Processor ABI Porting and Conformance Guide, drafts available from SGI.

X Window System: the Complete Reference to Xlib, X Protocol, ICCCM, XLFD, Robert W. Scheifler and James Gettys, with Jim Flowers and David Rosenthal, 3rd Edition, Digital Press, ISBN 1-55558-088-2. This is the primary X11R5 reference.

OpenGL Optimization and Porting Guide, available from SGI:

OSF/Motif Programmer's Guide, Revision 1.2, Open Software Foundation, Prentice Hall 1992, ISBN 0-13-643107-0. These two documents are the primary OSF/Motif 1.2 references.

OSF/Motif Programmer's Reference, Revision 1.2, Open Software Foundation, Prentice Hall 1992, ISBN 0-13-643115-1.