XL Compiler Support for New RISC System/6000 Architectures

By Jaime Vazquez and Ahmed Chibib

This article describes how IBM's XL compilers support the POWER, POWER2, and PowerPC Architectures. It discusses the trade-offs of optimizing the compile for one architecture versus compiling the application to work with all architectures.

Applications are the most critical element in commercial acceptance of new hardware platforms. Maintaining binary compatibility with existing applications is essential as system hardware and software evolve. Since its introduction in June 1990, the RISC System/6000 has attracted thousands of applications. IBM has maintained binary compatibility with these applications despite many significant hardware and software enhancements.

Until recently, all RISC System/6000 products were based on the POWER Architecture and its two implementations: RIOS1 (RS1), which is a multichip implementation, and the RISC Single Chip (RSC). Both implementations support the same instruction set and provide binary compatibility for all AIX applications.

On September 21, 1993, IBM announced new workstation and server products (Models 250, 58H, 590, and 990) based on two architectures that evolved from the original POWER Architecture:

- POWER2: A high-performance multichip follow-on to the POWER Architecture
- PowerPC: An architecture developed jointly with Motorola and Apple

RISC System/6000 models built in the future will be based on these new architectures. The PowerPC will initially be used in low-end workstations and servers while the POWER2 is targeted for high-end workstations and servers.

May '93
p.64

Jaime Vazquez

The following sections highlight the differences among the three major architectures and their implementations, including the impact these differences will have on binary compatibility and performance. Various compiler options that optimize for the different architectures are discussed.

POWER and POWER2 Architectures

The POWER2 Architecture is a superset of the POWER Architecture used in all RISC System/6000 products before October 1993. POWER2 is the basis for two deskside models: the 58H running at 50 MHz and the 590 running at 66 MHz. It is also used in the 990 rack model running at 71.5 MHz.

POWER implements one floating-point, one integer, and one branch unit. It can execute four instructions per cycle (five operations). The multichip POWER2 implements dual-integer and floating-point units and a branch unit that can execute six instructions per cycle (eight operations). The instruction capability enhances the performance of POWER2-based systems that are targeted for high-end desktop, deskside, and rack models.

The following are some other differences of the POWER2 compared to the POWER Architecture:

- Eleven new instructions
- Different page-table format—hash instead of inverted page tables
- Page aliasing
- Support for floating-point imprecise mode
- Different interrupt mechanism
- Faster clock speed

- Double the bandwidth from memory to the cache
- Double the bandwidth from the data cache to the integer and floating-point units
- Different alignment requirements for the quadword data
- A special data address break-point register
- A performance monitoring facility

PowerPC Architecture

The PowerPC Architecture allows processor designers the freedom to make hardware implementation trade-offs that have little or no effect on software development. This flexibility allows implementers to produce different PowerPC chips that vary in complexity, speed, power consumption, number of execution units, cache sizes, multiprocessor capability, and so on, without affecting the binary compatibility of the applications that will run on them.

The architecture allows for future extensions, such as larger cache sizes and additional execution units, while also providing processor designers latitude in some areas. It also provides a stable instructional base for applications and operating systems. The relaxed requirements allow chips to be faster, smaller, and cheaper than expected.

Some changes to the POWER Architecture contained in the PowerPC Architecture are as follows:

- 64-bit full architecture and 32-bit subset architecture
- Single-chip implementation
- Multiprocessor enabled
- Relaxed memory order (support for big-endian and little-endian)
- Scalability and extensibility (allows for different levels of price/performance and configurations)
- Implementation-specific and optional features
- Support for single-precision floating-point operations
- Support for L2 caches
- Omission of several POWER instructions
- Addition of several new 32-bit and 64-bit instructions
- Different I/O structure
- Different alignment requirements and interrupt handling
- Different real-time clock
- Support for big-endian and little-endian byte ordering

Compilation Modes

Figure 1 shows the relationship between the instruction sets of the various architectures and implementations. All existing applications use the POWER instruction set. Since the POWER2 instruction set is a superset of POWER, all existing RISC System/6000 applications will run unchanged on a POWER2-based system. Existing applications should run faster since POWER2 machines have a faster clock, larger data cache, and dual integer and floating-point units.

The additional instructions in the POWER2 instruction set will be generated by compilers only if the `-qarch=pwrx` flag is used. In this case, if any of the POWER2-specific instructions are generated, the application will run only on a POWER2-based system. It should be noted that the new POWER2 instructions have no impact on integer performance. Generally, only applications that manipulate double-precision floating point data will benefit from them.

The PowerPC Architecture defines additional 64-bit instructions for manipulating 64-bit integers and pointers. Applications that have 64-bit data can be specifically targeted for the 64-bit implementation (the 620 chip). They will run only on 64-bit PowerPC processors.

Figure 2 shows the relationship between the various compilation modes and the hardware on which the resulting binaries can correctly be executed.

Compiler Flags

The IBM XL compilers (IBM XLC Version 1.3, IBM C Set++ Version 2.1, IBM XL FORTRAN Version 3.1, and XL Pascal Version 1.1.3), as well as all future 1994 releases of these IBM compilers, will have several new flags that specify the target hardware for generating code. The new flags are as follows:

-qarch=pwr: Targets code generation to the POWER instruction set. Applications compiled with this flag will run on the POWER, POWER2, and the PowerPC 601 chip without emulation. However, software emulation may be required when running on future PowerPC implementations such as the 603, 604, and 620.

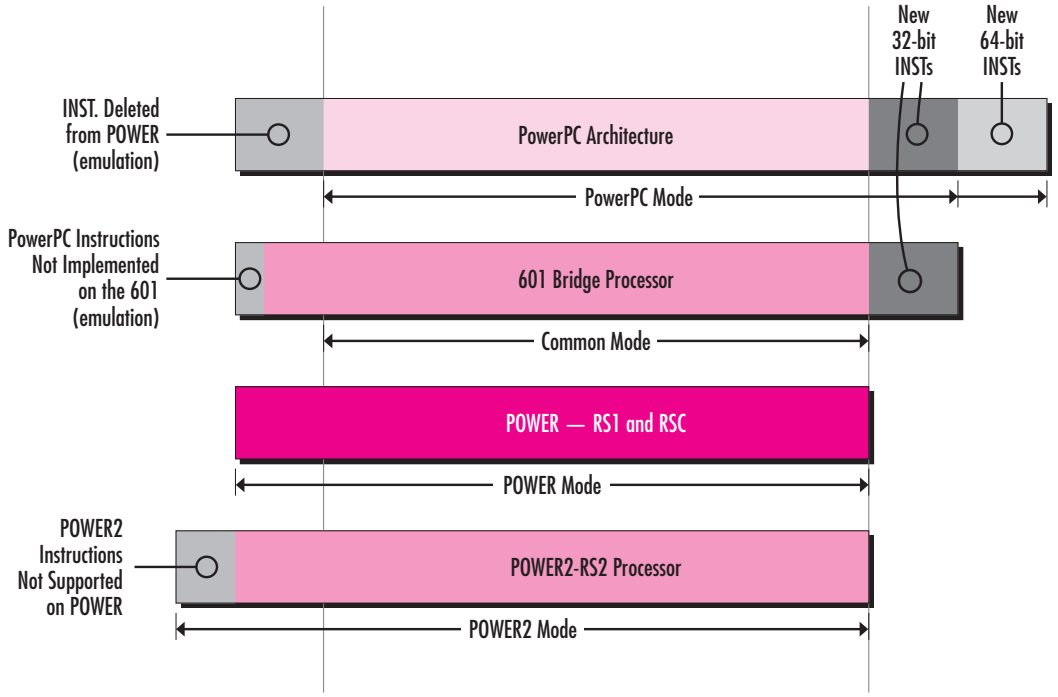

Figure 1. POWER, POWER2, and PowerPC instruction sets

-qarch=pwrx: Targets code generation to the POWER2 instruction set. Applications compiled with this flag will only run on POWER2-based systems if any of the 11 unique RS2 instructions is generated. There is no software emulation provided for these instructions on any other hardware.

-qarch=ppc: Targets code generation to the PowerPC instruction set. Applications compiled with this flag will run only on PowerPC-based systems with the 601, 603, 604, or 620 chipset. There is no software emulation provided for the new PowerPC instructions on POWER- or POWER2-based systems.

-qarch=com: Targets code generation to the subset of instructions common to both POWER and the PowerPC. Applications compiled with this flag will run on all RISC System/6000 platforms regardless of the processor being used. This mode guarantees binary compatibility across old, new, and future hardware without emulation. It is the recommended mode of compilation because it allows software developers to have a single binary of their application that will be suitable for all RISC System/6000s. Compiling with this flag can result in a slight performance degra-

Code Can Run On	Source Compiled for				
	Common	**POWER**	**POWER2**	**PowerPC 32 (32-bit)**	**PowerPC 64 (64-Bit)**
POWER	OK	OK	—	—	—
POWER2	OK	OK	OK	—	—
PowerPC 601	OK	OK	—	OK	—
PowerPC 32 (32-bit)	OK	OK (emulation)	—	OK	—
PowerPC 64 (64-bit)	OK	OK (emulation)	—	OK	OK

Figure 2. Binary compatibility for compiler modes

dation depending on the type of application and the hardware on which it is executed.

In `-qarch=com` mode, the compiler avoids generating instructions that are unique to a single architecture. In addition, for each integer multiply, divide, and remainder operations, the compiler generates a `Branch and Link Absolute` instruction. The binder resolves this instruction during the linking phase to a known low-memory location that contains the proper sequence of instructions to perform the particular operation. These code sequences, called *millicode*, are unique for each architecture. The system booting process initializes these low-memory locations with the proper set of millicode routines based on the architecture of the machine being booted. There are six millicode routines in low-memory beginning with AIX Version 3.2.3. Releases before AIX 3.2.3 do not have the millicode routines and cannot execute programs that branch and link to these routines.

The IBM XLC 1.3, XL C Set++ 2.1, XL FORTRAN 3.1, and XL Pascal 1.1.3 compilers default to a partial, common mode of compilation. In this mode, the XL compilers avoid generating all architecture-unique instructions. They do not generate branch and link to millicode routines for integer multiply and divide. Instead, they will generate the POWER instructions for these operations, which allows applications compiled in this partial common mode to run on operating systems earlier than AIX 3.2.3 that do not have the millicode routines.

The AIX 3.2.5 kernel, as well as all libraries, commands, and subsystems, are compiled with `-qarch=pwr`. Future releases of the XL compilers will support a default mode that is full common mode, including the calls to the millicode routines. AIX Version 4 libraries will be compiled in this mode.

Other Compiler Options

The superscalar implementations of the POWER, POWER2, and PowerPC Architectures have multiple execution units. Improved execution speed can be obtained by rearranging the sequential order of instructions into parallel instruction streams that can be executed by several execution units. This type of optimization is called *scheduling*. Since the number of execution units and their pipeline depth varies from one implementation to the next, the ability to specify which target CPU the compiler favors for scheduling is important. Specifying the target CPU is done by using the `-qtune` compiler flag.

In general, tuning the scheduling for one implementation versus another will not significantly change performance on the target CPU, nor will it significantly degrade the performance on the others.

The following list describes the compiler instructions for various architectures.

-qtune=pwr: Directs the compiler to schedule instructions favoring the POWER implementation.

-qtune=pwrx: Directs the compiler to schedule instructions favoring the POWER2 implementation.

-qtune=601: Directs the compiler to schedule instructions for the PowerPC 601 implementation.

In the future, additional options will be available to specify tuning for the other PowerPC implementations (603, 604, and 620 chips).

If the `-qarch` option is set to `pwr`, `pwrx`, or `ppc`, the `-qtune` option will default to `pwr`, `pwrx`, or `ppc` respectively. If the `-qarch=com` option is set explicitly or by default, the `-qtune` option will default to `pwr`. Figure 3 shows the valid combinations of the `-qarch` and `-qtune` options.

-qarch=	-qtune=		
	pwr	**pwrx**	**601**
com	x	x	x
pwr	x	x	x
pwrx	x	x	
ppc	x		x

Figure 3. Combinations of valid -qarch and -qtune

Hybrid Mode of Compilation

Most applications, particularly those with little or no floating-point computations, should be compiled in common mode. The advantages of having a single binary outweigh the small performance degradation that may result from compiling in common mode. However, certain double-precision floating-point applications can benefit from targeting POWER2 (`-qarch=pwrx` option), while other single-precision floating-point applications can benefit from targeting the PowerPC (`-qarch=ppc` option). Of course, targeting for a specific architecture will either restrict the application from running on some processors or may negatively affect its performance on other processors—both of which are undesirable consequences.

The hybrid compilation mode allows a software developer to compile performance-critical routines in more than one mode and provide

runtime testing to determine which copy of the routine to execute based on the type of hardware used by the application.

Migration of POWER Applications to POWER2

All applications compiled on the original POWER Architecture will run unmodified on the POWER2 platforms without recompilation. The only change is that performance will increase substantially. The performance increase depends on how efficiently the additional execution units are being used.

Enhanced performance can be obtained by recompiling and retargeting the application for the POWER2 Architecture. However, any application recompiled for POWER2 processors can run only on POWER2 machines.

Performance degradation can be avoided either by compiling explicitly for the PowerPC Architecture or by compiling using the common-mode flag.

A shared library version of IBM's Engineering and Scientific Subroutine Library (ESSL) is platform-specific to POWER2. Beginning with AIX 3.2.5, all compilers, the assembler, and the debuggers support the POWER2 Architecture.

Migration of POWER Applications to the PowerPC 601

The PowerPC 601 processor is a bridge between existing POWER and PowerPC processors because it can support all POWER instructions that are generated by the compilers. The operating system masks most of the hardware differences. Existing compiled applications will run at full speed without instruction emulation. Five of the original POWER instructions emulated by the 601 processor will run more slowly. However, three of the five emulated instructions are for cache operations, of which one is a privileged instruction and none are generated by the compilers.

The performance of single-precision floating-point applications will improve if they are compiled for the PowerPC. However, applications compiled explicitly for the PowerPC will not run on POWER and POWER2 processors. Tools available in the future will identify the presence of deleted, unsupported, or emulated instructions in existing applications.

Migration of POWER to PowerPC 603/604/620

The PowerPC supports most POWER user-mode instructions. Unsupported instructions will trap and cause emulation code to run. The emulation of POWER instructions will negatively affect performance. Performance degradation can be avoided either by compiling explicitly for the PowerPC Architecture or by compiling using the common-mode flag.

When compiled explicitly for the PowerPC Architecture, applications will run only on that architecture. Applications compiled with the common mode will run unmodified on all the architectures, although performance may suffer.

Common Mode and Performance on Different Architectures

When running applications on POWER models, performance degradation, in most cases, should be minimal. Few of the deleted POWER instructions are generated by compilers, and most of those are low-frequency instructions (occur less than 1% in normal compiled code). In most cases, three to four instructions from the common set can be substituted. Applications running on POWER2 behave similarly. Performance of certain double-precision applications can be improved if targeted for POWER2 models.

Applications compiled in common mode running on PowerPC processors will perform more slowly if they use extensive single-precision floating-point computation. Double-precision floating-point operations take longer than single-precision operations. This performance degradation is caused by computations on single-precision floating-point data being done in double-precision and then converted back to single-precision floating point. In addition, compiling in common mode prevents the compilers from generating new PowerPC instructions, such as floating-point square root (`fsqrt`) and extended sign byte (`extsb`). The overall performance degradation of software applications will vary based on the amount of time spent doing single-precision computation.

Jaime Vazquez, *IBM Corporation, RISC System/6000 Division, 11400 Burnet Road, Austin, TX 78758. Internet: jaime@austin.vnet.ibm.com. Mr. Vazquez currently is working in the OEM Consulting Services department. He has a Master of Computing Sciences degree from Texas A&M University.*

Ahmed Chibib, *IBM Corporation, RISC System/6000 Division, 11400 Burnet Road, Austin, TX 78758. Internet: chibib@austin.ibm.com. Mr. Chibib is a senior programmer responsible for providing technical assistance to software developers. He has been a member of the PL8 compiler development group for the IBM RT RISC processor and was involved with the definitions of the AIX linkage conventions for the RISC System/6000 and the low-level application binary interface for the PowerOpen environment. He is currently working on binary*