The IBM 6x86 and 6x86L Microprocessor Architecture

Delivering Next-Generation Performance on Today's Installed Base

®

Application Note

Revision Summary: This revision contains minor corrections and changes in terminology.

PROCESSOR
Manufactured by IBM

January 17, 1997

Fax #40200

Introduction

As the desktop PC takes on more and more sophisticated tasks, such as multimedia, imaging, high-end desktop publishing and statistical modeling, the need for processing power increases dramatically. To meet that demand, a new generation of microprocessors is entering the marketplace. Designed to deliver processing power that is many times greater than the venerable 486, these processors utilize architectures that differ substantially from that of their x86 predecessors. These new processor architectures fall into two broad groups, those that have adopted a RISC-based architecture, and those that are pushing the x86 performance envelope.

The RISC-based architectures use a reduced instruction set to achieve their performance. To get the promised performance out of the RISC-based architectures, users must run software applications written and optimized for that architecture. Existing x86 applications must be run in emulation mode, which reduces performance. These requirements, together with the additional instructions present in the x86 architecture, reduce the price/performance margin of these RISC architectures.

Intel's Pentium® is an evolution of Intel's x86 family, utilizing a superscalar architecture with multiple integer units to achieve its performance. As a development of the x86 family, it avoids the pitfall of the RISC-based architectures by remaining compatible with applications compiled for the x86 instruction set. However, because it relies on instruction reordering by a compiler, the performance gain is lost to any applications not compiled specifically for the Pentium.

The IBM 6x86 and 6x86L processor architectures are a revolutionary advancement of previously available x86 processors, utilizing both superscalar and superpipelined technologies to achieve high performance. Like the Pentium processor, it is fully x86 instruction set compatible. Unlike the Pentium processor, the IBM 6x86 and 6x86L processors implement a complex dependency removal scheme that allows it to deliver many times the performance of the 486 processor. This performance margin represents a gain of 30% over the Pentium processor at identical clock rates when running from the installed base of expensive software, giving the IBM 6x86 and 6x86L processors a wide price/performance margin over the Pentium and RISC-based architectures.

High-Performance Processor Architectures

Simply driving up the clock rate on a 486 architecture cannot achieve the kind of significant performance increase necessary to meet the demands of the new marketplace. The conventional x86 microprocessor architecture presents too many execution obstacles to allow major performance gains from clock speed increases alone. Instead, the basic architecture of the microprocessor must be changed to improve instruction execution and allow the processing units to take full advantage of high clock rates. The two basic architectures which provide this kind of performance increase are the superscalar architecture and the superpipelined architecture.

Superscalar processor architectures use instruction parallelism - executing two separate instructions at the same time - to reduce the amount of time the processor spends executing the series of instructions required to complete a task. This provides the potential of doubling performance at any given clock rate, as it effectively adds another processor running in parallel.

Superpipelined architectures take the conventional pipelined architecture, where the instruction execution cycle is broken up into several distinct stages that perform particular tasks, and increase the number of stages. By refining the tasks executed in each stage and implementing more stages, the processor can take full advantage of very high clock rates. Superpipelining doubles the processor's clock rate providing the potential for doubling processor performance.

The optimal performance increase is achieved by combining the superscalar and superpipelined architectures into a single architecture, where two instructions are executed in parallel superpipelines at very high clock rates.

High-Performance Obstacles

If achieving high performance processors were merely a matter of designing a superscalar, superpipelined architecture running at very high clock rates, all microprocessors would be essentially the same. Unfortunately, the issues involved in implementing superscalar, superpipelined architectures present a number of complex barriers to maximizing the potential of those architectures.

Control and data dependencies represent a significant barrier to efficient use of the superscalar or superpipelined architectures. With two instructions executing in parallel, any dependency between the two instructions can result in a stall, where the dependent instruction is stuck in the pipeline until the data or resource it requires becomes available. Instruction dependencies can include any situation where one instruction requires the operand, result, or data from an instruction executing near it in the pipeline. While an instruction is stalled in a pipeline, a bubble forms below it, as portions of the pipeline are left unused for a number of clock cycles. Stalls greatly reduce the performance of superscalar, superpipelined architectures.

Change of program flow (COF) instructions result in additional performance degradation in superscalar or superpipelined architectures. If an instruction is a change of flow instruction such as a jump, conditional branch, or call instruction, and the result of that instruction causes the program to need an instruction other than the one immediately following the current stream, the current instruction must be cleared and the new instructions fetched. All of this requires time, and effectively wastes the pipeline, which reduces the performance of the processor, especially in superpipelined or superscalar architectures.

Resource conflicts are also detrimental to the performance of superscalar or superpipelined architectures. The x86 architecture uses eight logical registers. If two instructions are executing down identical pipelines, there is a chance they will require access to the same register at the same time, creating a resource conflict. This situation results in a stall while one instruction waits to access the contested register. The stall creates a bubble, just as in control and data dependencies.

Problematic Solutions

To maximize the use of a superscalar or superpipelined microprocessor, dependencies and resource conflicts must be eliminated, and a mechanism for handling COF without degrading performance must be implemented. There are a number of solutions to these issues, but the solutions are not without problems of their own.

New Software that replaces or adds to the existing installed base of software with applications designed specifically for the new architecture ensures optimal use of the available processing power. New software eliminates many of the conflicts resulting from dependencies, COF, and resource conflicts because it utilizes a new tool set. This approach off-loads the responsibility for maximizing the performance of the new architecture from the microprocessor developer onto the third-party software vendor. The third-party vendor then passes as much of the development burden as possible on to the end-user with an increase in total system cost. The price/performance margin of this type of solution severely limits the success of unique architectures. Additionally, software developed for unique architectures loses its appeal by its incompatibility with the millions of users who already have x86-based PCs.

Recompilation of existing software and hardware achieves much of the same benefit as replacement, as it allows for optimizing the source code to work with a specific processor. The optimization process removes many dependencies and resource conflicts, as well as fine-tuning COF controls. However, recompilation again shifts the burden from the processor developer onto the third-party software or compiler developer and the end-user. The time and resources committed to recompilation, including the necessary QA, repackaging, and marketing efforts drive down the price-performance margin and reduce the attractiveness of the new architecture to the end-user, who must still purchase software upgrades in order to achieve the promised performance. The hidden cost of software upgrades can easily add thousands of dollars to the end-user's cost. This is the path that was chosen for the Pentium processor.

Emulation of the x86 architecture is a path followed by some RISC-based architectures to gain access to the installed base of software without recompilation. Emulation cannot match the performance of execution in x86 native mode. New architectures typically lose much of their performance gain over previous designs when they are forced to run in emulation mode. While the price/performance margin of the machine itself is attractive, the performance achievable by the end users with their existing software is reduced when running in emulation mode. The cost of developing software to run on the machines further reduces the attractiveness of these new architectures.

Architectural enhancement is the path that has been chosen for the 6x86 processor family. Making architectural enhancements to the new processor design provides all of the advantages of recompilation dependency removal, reduction of resource conflicts and better COF control without raising compatibility barriers or off-loading the development burden to software vendors and the end-user. This approach provides access to the installed base of software while delivering improved performance at reasonable cost. This also delivers significant performance increases with existing software, without delays in availability and additional costs associated with software upgrades.

A High-Performance Solution

To solve the problems involved in running the installed base of software and the upcoming generation of applications on a superscalar, superpipelined architecture requires more than an edict for recompilation or emulation. What is needed is a microprocessor combining the existing architecture with built-in hardware to eliminate or reduce dependencies and resource conflicts, while still delivering high performance at a competitive price. The IBM 6x86 and 6x86L processor architecture achieves this solution.

Data dependencies are nearly eliminated. Dynamic, transparent register renaming, combined with an extended set of 32 general purpose registers, nearly eliminates resource conflicts. The IBM 6x86 and 6x86L processor's register renaming scheme removes both Write-After-Read (WAR) and Write-After-Write (WAW) dependencies, and allows the resolution of Read-After-Write (RAW) dependencies.

Data forwarding removes data dependencies between instructions normally requiring serialization. By forwarding the operand or result from the "leading" instruction to the "following" instruction, data forwarding allows the instructions to execute in parallel. This eliminates serialization stalls. Data forwarding removes most RAW "true" dependencies. Data bypassing allows a memory or register operand to be passed directly to the next instruction, without waiting for the memory location or register to be updated, further reducing RAW "true" dependencies.

Control dependencies are also reduced. Branch prediction speeds instruction execution by predicting the program flow change that will result from a conditional branch instruction. This eliminates the need to wait until a COF instruction has executed before fetching the next instruction. Provided the predictions are correct, the processor can keep the pipeline full. If the prediction is incorrect, however, the pipeline must be cleared of the incorrect instruction before the correct instruction can be executed. Branch prediction, by itself, provides a performance increase at some risk.

Speculative execution greatly enhances the benefits of branch prediction. Speculative execution allows the processor to execute the predicted instruction flow as if it were the correct one. The IBM 6x86 and 6x86L processor architecture goes beyond conventional speculative execution by continuing to predict COF branches and fetching instructions based on those predictions. The IBM 6x86 and 6x86L processor can speculatively execute past four COF branches or floating point operations without stalling, while maintaining precise exception for floating point operations. The risk still exists that the predicted branch will be the wrong one. If that is the case, the processor must not only repair the branch instruction, but also any speculatively executed instruction. Any intermediate results must also be undone or repaired. The IBM 6x86 and 6x86L processor architectures implement a sophisticated repair scheme which eliminates this.

Transparent repair of prediction errors allows the processor to clear the pipeline of the incorrect instruction and resume processing on the correct instruction in the amount of time required to clear a single branch prediction made without speculative execution. This means that the IBM 6x86 and 6x86L processor can reap the benefits of speculative execution of up to four COF branches while incurring no more penalty than the more pedestrian architectures that merely guess at COF results. These branch control schemes mean that the IBM 6x86 and 6x86L processor architecture achieves a throughput of one clock cycle for correctly predicted branches, while incurring no additional penalty for incorrect predictions.

Optimal pipeline utilization is achieved by two methods, one at instruction issue and one at instruction execution. Intelligent instruction issuing minimizes bubble formation and allows for bubble squashing. In most advanced architectures, an instruction that has an inter-instruction dependency may be stalled by the processor at issue time, before it stalls on its own in the pipeline. With intelligent instruction issuing, instructions are allowed to move as far down the pipeline as possible, until they stall as a result of a dependency or an instruction stalls ahead of them. With the architecture's advanced mechanisms to dynamically remove dependencies, this means that most stall conditions never actually occur.

In addition, only a small group of instructions has any issuing constraints. These include COF, floating point, and exclusive instructions, which are restricted to using the X-pipe. Change of flow and floating point instructions allow integer instructions to use the Y-pipe while they execute in the X-pipe. Exclusive instructions utilize the processing power of both pipelines to accelerate execution. Exclusive instructions are those that may fault during execution, typically those involving multiple memory accesses. In addition, intelligent instruction issuing implements load balancing between the X and Y pipelines in an effort to minimize any dependencies.

Out of order execution allows instructions to complete independently of each other, thereby preventing a stall in one pipeline from forcing a stall in the other. Bus cycles are issued and exceptions are generated in program order.

The Right Combination

The IBM 6x86 and 6x86L processor architecture is a superscalar, superpipelined architecture capable of operating at very high clock rates. The architecture's sophisticated schemes for reducing dependency and conflict are implemented in hardware, allowing it to deliver performance increases of roughly 2.5 times that of the 486 architecture operating at an identical clock rate, and a gain of 40% - 60% over the Pentium processor at an identical clock rate when running today's unrecompiled applications. This architectural advantage, coupled with the aggressive clock rates targeted for the IBM 6x86 and 6x86L microprocessor, yields up to 5x the performance of a 486-50, and up to 2 times that of a current Pentium processor.

The IBM 6x86 and 6x86L processor architecture also provides more than twelve times the performance of a typical RISC-based architecture running existing x86 software in emulation mode. The IBM 6x86 and 6x86L processor architecture will enjoy a wide price/performance margin over the Pentium processor and RISC-based architectures.

The IBM 6x86 and 6x86L Microprocessor Architecture

The IBM 6x86 and 6x86L processor architecture is a high performance x86 processor architecture bridging the need for performance and compatibility with today's 16-bit applications to the potential of tomorrow's 32-bit applications. The IBM 6x86 and 6x86L processor architecture comprises five basic elements:

- Integer Unit
- Cache
- Memory Management Unit
- Bus Control
- Floating Point Unit

Integer Unit

The IBM 6x86 and 6x86L processor architecture utilizes two, seven-stage integer pipelines, the X-pipe and the Y-pipe. Each pipeline comprises a prefetch stage, two decode stages (D1, D2), two address calculation stages (AC1, AC2), an execute stage, and a write-back stage.

Prefetch fetches up to 16 bytes (four instructions) per bus clock. Branch Target Buffer (BTB) checks and branch predictions are performed during the prefetch cycle to fetch instructions from the predicted path.

Decode determines the instruction length (D1), decodes the instruction, and determines the optimal pipeline in which to execute the instruction (D2).

Address calculation calculates up to two effective addresses per clock cycle, performs register renaming, and makes scoreboard checks (ACl). The second address calculation stage (AC2) performs Translation Lookaside Buffer (TLB), cache, and register file accesses, as well as performing segmentation and paging checks.

Execution performs ALU operations and MUL/DIV instructions.

Write-back writes to the register file and write buffers, and updates the machine state.

Cache

The IBM 6x86 and 6x86L processor architecture contains a large, on-chip, 4-way set associative, unified instruction/data cache as the primary data cache and secondary instruction cache, and a 256-byte, fully set associative, instruction line microcache that is the primary instruction cache. When coupled with the instruction line microcache, this architecture provides a greater hit rate over a wider percentage of applications than architectures dedicating half the cache to instructions and half to data.

The unified cache is dual-ported to allow for two simultaneous fetches, reads, writes, or combinations of any two. Again, when coupled with the primary instruction line microcache, this gives the IBM 6x86 and 6x86L processor architecture a cache bandwidth similar to Harvard-style caches while retaining the advantages of the superior hit rate.

Memory Management Unit

The IBM 6x86 and 6x86L processor architecture is fully x86 compatible, so the memory management unit adheres to the standard segmentation and paging mechanisms.

Bus Controller

The IBM 6x86 and 6x86L processor architecture logically isolates the bus control unit. This isolation provides for wide product flexibility in order to address specific market needs.

Floating Point Unit

The IBM 6x86 and 6x86L processor architecture contains a built-in, 64-bit, enhanced x87-compatible floating point unit. The FPU is IEEE-754 compatible and utilizes the x87 instruction set.

The IBM 6x86 and 6x86L processor architecture contains a single floating point pipeline, enhanced by a four-instruction queue and independent, 64-bit write buffers. This scheme provides several key advantages. Floating point operations appear as a single-clock instruction to the integer pipeline and, while they themselves must traverse the X-pipeline, they allow any integer instruction to be simultaneously paired in the Y-pipeline, providing parallel execution. The processor architecture also allows out-of-order execution of integer instructions relative to floating point instructions, and speculative execution of up to four floating point instructions. Because floating point instructions are speculatively executed, precise exceptions - an x86 software compatibility requirement - can be maintained without creating stalls due to multiple floating point instructions. The following table lists some comaprisons:

FEATURE	IBM 6x86 and 6x86L PROCESSOR	INTEL PENTIUM PROCESSOR
x86 Instruction Set	x	x
Superscalar	x	x
Multiple Integer Units	x	x
Superpipelined	x	
GP Registers	32	8
Register Renaming	x	
Data Forwarding	x	
Branch Prediction	x	x
Speculative Execution	x	
Out-of-Order Execution	x	

Table 1. Feature Comparison

Summary

The IBM 6x86 and 6x86L processor architecture is a revolutionary advancement of the x86 family of microprocessors, utilizing both superscalar and superpipelined technologies to achieve high performance. The architecture not only overcomes such performance barriers as data dependencies, change of flow instructions and resource conflicts, but also brings the end user a superior performance advantage without the added cost of purchasing recompiled software. As the desktop PC takes on more and more sophisticated tasks, IBM continues to provide micro-processor solutions that bring cost effective power to the end user.