|  |
| --- |
|  |
| D:\project\Internship Reports\BDU1.jpg |
| BahirDar University |
| Institute of Technology |
| School of Computing And Electrical Engineering |
| Department Of Computer Science And Engineering |
|  |
| **Atnatios** |
| **[Pick the date]** |

|  |
| --- |
| [Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.] |

# Table of Contents

[Table of Contents 2](#_Toc295206407)

[1. Introduction of PowerPC 5](#_Toc295206408)

[2. IBM POWER3 System Micro Architecture 6](#_Toc295206409)

[2.1. Major design constraints and challenges 7](#_Toc295206410)

[2.2. Overview of the POWER3 processor 7](#_Toc295206411)

[2.2.1. Instruction processing unit and instruction flow unit 9](#_Toc295206412)

[2.2.2. Fixed-point execution units 11](#_Toc295206413)

[2.2.3. Floating-point execution units 11](#_Toc295206414)

[2.2.4. Load/store execution units 13](#_Toc295206415)

[2.2.5. Data cache unit 13](#_Toc295206416)

[2.2.6. Memory management unit 14](#_Toc295206417)

[2.2.7. L1 data cache 14](#_Toc295206418)

[2.2.8. Data pre-fetching 15](#_Toc295206419)

[2.2.9. Bus interface unit 16](#_Toc295206420)

[2.2.10. L2 cache 16](#_Toc295206421)

[3. IBM Power4 System Micro Architecture 17](#_Toc295206422)

[3.1. Chip Features 17](#_Toc295206423)

[3.2. Processor Features 17](#_Toc295206424)

[3.3. Performance enhancement techniques 18](#_Toc295206425)

[3.3.1. Pipeline Requirements 18](#_Toc295206426)

[3.3.2. Pipeline Improvements 18](#_Toc295206427)

[3.3.3. Load/Store Unit Operation 18](#_Toc295206428)

[3.4. Memory Hierarchy 19](#_Toc295206429)

[Cache Coherence 19](#_Toc295206430)

[4. IBM Power5 System Micro architecture 19](#_Toc295206431)

[4.1. System feature 20](#_Toc295206432)

[4.2. Processor Characteristics 20](#_Toc295206433)

[4.3. Micro partitioning 21](#_Toc295206434)

[4.3.1. Multi Chip Module (MCM) Architecture 21](#_Toc295206435)

[4.3.2. Registers 21](#_Toc295206436)

[4.4. Performance enhancement technique 21](#_Toc295206437)

[4.4.1. Chip Stats 21](#_Toc295206438)

[4.4.2. Pipeline 22](#_Toc295206439)

[4.4.3. Instruction Fetch 22](#_Toc295206440)

[4.4.4. Branch Prediction 22](#_Toc295206441)

[4.4.5. Instruction Grouping 22](#_Toc295206442)

[4.4.6. Performance Expectations 23](#_Toc295206443)

[5. IBM POWER6 System Micro Architecture 23](#_Toc295206444)

[5.1. Introduction 23](#_Toc295206445)

[5.2. Description 23](#_Toc295206446)

[5.3. High-frequency core design 25](#_Toc295206447)

[5.4. Balanced system throughput 27](#_Toc295206448)

[5.5. Processor core 29](#_Toc295206449)

[5.6. POWER6+ 32](#_Toc295206450)

[5.7. Products 32](#_Toc295206451)

[5.8. Summary 33](#_Toc295206452)

[6. IBM PowerPC 620 System Micro architecture 34](#_Toc295206453)

[6.1. Abstract 34](#_Toc295206454)

[6.2. Introduction 34](#_Toc295206455)

[6.3. The PowerPC 620 Architecture 35](#_Toc295206456)

[6.4. Machine Specification 39](#_Toc295206457)

[6.5. Experimental Framework 40](#_Toc295206458)

[6.6. PowerPC 620 Instruction Fetching 40](#_Toc295206459)

[6.7. PowerPC 620 Instruction Dispatching 40](#_Toc295206460)

[6.8. Improved Out-of-Order Execution 40](#_Toc295206461)

[6.9. Two-Level Cache Structure Added 41](#_Toc295206462)

[6.10. Memory Bandwidth Increased 41](#_Toc295206463)

[6.11. Cost of Complexity Is High 41](#_Toc295206464)

[6.12. Summary 42](#_Toc295206465)

[7. IBM PowerPC 970 System Micro Architecture 43](#_Toc295206466)

[7.1. Characteristics of PowerPC 970 43](#_Toc295206467)

[7.2. Performance Enhancement Technique 45](#_Toc295206468)

[7.2.1. Altivec - Velocity Engine - Vector SIMD 45](#_Toc295206469)

[7.2.2. Pipeline Depth 46](#_Toc295206470)

[7.2.3. Preliminary Specifications 46](#_Toc295206471)

[7.2.4. Packet Based System Interconnect 47](#_Toc295206472)

[7.2.5. 32 Bit and 64 Bit Architecture 47](#_Toc295206473)

[7.2.6. System Chip Support 48](#_Toc295206474)

# Introduction of PowerPC

The first PowerPC is design based on the IBM’s POWER (Performance optimization with enhanced RISC) architecture used in IBM’s RISC system/6000. It inherits the characteristic of POWER architecture that incorporated fundamental RISC features of fixed width instruction set, register to register architecture, a few simple memory-addressing modes, and simple hardwired instructions. It is designed with the aim to minimize the total time required to complete a task. The total time is the product of 3 components:

* Path length
* Number of cycles needed to complete an instruction, and
* Cycle time

The unique feature of the PowerPC is that, it defines a 64bit architecture that is a superset of the 32bit architecture, providing application binary compatibility for the 32 bit applications. This approach assures upward compatibility for all applications.

The Motorola PowerPC 601 is the first implementation of the PowerPC family of reduced instruction set computer (RISC) microprocessors. The architecture aims to improve performance by exploiting instruction level parallelism by incorporating more resource in specific areas. It supports out of order execution with the multiple independent EUs. The architecture integrates three execution units to increase the level of parallelism. Each EU having their own set of register resources to reduce the communication and synchronization required between the EUs. The instruction unit contains an instruction queue of 8 instructions that is refilled in a single cycle from the instruction cache. The lower half of the instruction queue is analysis by the BPU for any potential branches. The BPU performs condition register (CR) look-ahead operations on conditional branches. Branch folding is used for to eliminate a branch instruction and issue the next instruction. Branch instructions that are executed are decoded but not issued for any further processing. The BPU attempts to resolve branches early which help to achieve the effect of a zero-cycle branch in many cases. In the case of a mis-predicted branch, the instruction fetcher flushes all predicted path instructions that are issued and reissue instruction from the correct path.

The pipeline structure in PowerPC demonstrates a fully interlocked instruction pipeline, which permits proper operation of the pipeline without introducing no-ops instructions for unused pipeline delay. The design helps to isolate the software from the dependence on the underlying hardware structure from future compatibility issues. In additional, it reduces program length, instruction bandwidth and cache entries.

Down here we try to discuss some of the micro architecture employed by the IBM Corporation on the PowerPC. Some of these lists of the PowerPC series are:

* PowerPC 2
* PowerPC 3
* PowerPC 4
* PowerPC 5
* PowerPC 6
* PowerPC 620
* PowerPC 970

# IBM POWER3 System Micro Architecture

The POWER3 processor is a high-performance microprocessor which excels at technical computing. Designed by IBM and deployed in various IBM RS/6000 systems, the Superscalar RISC POWER3 processor boasts many advanced features which give it exceptional performance on challenging applications from the workstation to the supercomputer level. In this paper, we described the micro architectural features of the POWER3 processor, particularly those which are unique or significant to the performance of the chip, such as the data prefetch engine, non-blocking and interleaved data cache, and dual multiply–add-fused floating-point execution units.

The IBM POWER3 processor is a 64-bit symmetric multiprocessing- enabled superscalar RISC microprocessor which is the heart of a new line of RS/6000\* workstations and server products. Designed to run at frequencies up to one-half gigahertz, the POWER3 processor supports even the most challenging technical computing applications.

This part describes:

* The motivation for the creation of the POWER3 processor.
* The challenges that it addresses.
* The highlights of its micro architecture that is significant to performance.

Product motivation and technological challenges

The POWER3 processor is the successor to the POWER2 processor; it was designed primarily to meet the demands for technical computing which come from a wide variety of customers in nearly every major sector of the market place: automotive, aerospace, pharmaceutical, weather prediction, energy, defense, electronics, chemical processing, bioengineering, environmental, and many areas of research. As with most modern microprocessor design efforts, the POWER3 processor required an enormous investment involving many highly skilled and talented professionals. Such an investment is prudent if and only if it addresses a large and growing market need, as is the case with modern technical computing.

## Major design constraints and challenges

The success of the personal computer has profoundly influenced the market for high-performance systems, with the main effect being an extreme sensitivity to price. Since much of the cost of a system is often associated with memory, a decision to use anything other than commodity DRAM, which enjoys the benefits of large scale manufacturing due to the personal computer boom, would drastically increase the price of adequately configured systems. But DRAM-based memories present a difficult challenge to designers attempting to build systems in which performance scales with increasing processor frequency. As processor frequencies have soared, memory latency has decreased only modestly, with the result being that memory latency, in processor cycles, has grown. At the same time, the demand for large parallel machines has increased physical bus length, protocol, and contention. The burden for overcoming these problems falls mainly on the microprocessor. Technical computing applications present computer designers with additional formidable challenges which center on floating-point and fixed-point computational speed, scalability, load/store bandwidth, and cache capacity. The POWER3 processor was designed to meet the rigors of technical computing applications, as well as the more general requirements of the high-performance computing market place, including reliability, ease of programming, addressability, and power and space restrictions.

## Overview of the POWER3 processor

The POWER3 processor is a CMOS-based superscalar RISC microprocessor which conforms to the PowerPC Architecture. Its two most fundamental architectural features, symmetric multiprocessing (SMP) systems and 64-bit effective addressing, provide the basic necessary for the contemporary challenges of technical computing. Multiprocessor systems not only increase the computing capacity of the system under a single image; they also allow applications to improve performance by exploiting shared-memory parallelism. The high level of performance that multiple POWER3 processors can produce, along with 64-bit program addressability, allows customers to solve larger three-dimensional simulations that were previously impractical. Because the POWER3 processor is a 64-bit implementation, it supports both the 32-bit and 64-bit modes provided by PowerPC.

The POWER3 processor has also been designed to span several advances in CMOS technology, allowing it to more than double its initial product frequency of 200 MHz over its product lifetime. To date, POWER3 processors have been shipped in RS/6000 products at frequencies ranging up to 450 MHz Applications which are optimized to the POWER3 platform can immediately take advantage of upgrades to faster POWER3 processors. The POWER3 processor is partitioned into seven functional blocks.

* Instruction processing unit (IPU).
* Instruction flow unit (IFU).
* Fixed-point unit (FXU).
* Floating-point unit (FPU).
* Load/store unit (LSU).
* Data cache unit (DCU).
* Bus interface unit (BIU).

Fixed-point unit

Load/store unit

Floating-point unit

Instruction flow unit

(IFU)

Instruction processing unit (IPU)

Data cache unit

(DCU)

FXU0

FXU1

FXU2

GP registers

LSU0

LSU1

FPU0

FPU1

FP registers

Bus interface unit

L2 cache 1–16 MB

Figure (1), POWER3 processor functional unit block diagram.

These functional units are discussed in the following sections.

### Instruction processing unit and instruction flow unit

Processor performance begins with the task of fetching the instructions for an application, partially decoding them, and dispatching them to the proper execution unit. The IPU and IFU are responsible for fetching, caching, and managing the flow of instructions during their tenure in the microprocessor (the tenure of a given instruction begins when it is dispatched to an execution unit and ends when it is completed). Logically, instructions are fetched from memory; however, for performance reasons, the IPU implements a 32-kilobyte (KB) instruction cache and a cache reload buffer (CRB). The instruction cache holds 256 cache lines, each of which is 128 bytes in length, and is organized as two 128-way set-associative arrays. The instruction cache provides single-cycle access. The CRB holds the most recent cache line transferred from memory. To provide support for virtual storage, a 256-entry two-way set-associative instruction translation look aside buffer (ITLB) and a 16-entry instruction segment look aside buffer (ISLB) are also implemented. The IFU attempts to keep as many instructions as possible executing in parallel in the machine, maximizing instruction throughput. Up to eight instructions are fetched per cycle, up to four are dispatched per cycle, and up to four instructions per cycle can be completed. To improve throughput, instructions are dispatched in order, most are allowed to execute and finish out of order, and then all instructions complete in order. (Architectural registers are updated only when an instruction targeting them completes). Executing and finishing instructions out of order increases the degree of instruction-level parallelism by allowing subsequent operations to execute both in parallel with logically prior long-running operations and ahead of operations which are delayed because of cache misses. Instructions are dispatched to the various functional unit instruction queues and are tracked with an entry in the 32-entry completion queue. These unit instruction queues ensure each functional unit an adequate supply of instructions from which to select for execution; they also provide a place for the instruction flow unit to place instructions so that a stalled instruction does not block dispatching of subsequent instructions. In many designs, dispatch bandwidth is a frequent bottleneck. The robust implementation of the POWER3 processor greatly reduces the likelihood that performance will be affected by dispatch restrictions. Since operand availability is not a requirement for dispatch, availability of space in the instruction queues and in the completion queue are the two primary restrictions on dispatch. The completion block ensures that the architectural state of the processor is always correct, enforcing in-order completion of committed instructions and ensuring that exceptions and interrupts are handled properly and in order. The POWER3 processor uses two mechanisms to improve branch-prediction accuracy. First, by tracking all outstanding condition-code-setting instructions, the CPU can determine when the branch outcome is known at dispatch, obviating the need to guess the direction of a branch. For branches that are unresolved at dispatch, the outcome is guessed and instructions are dispatched speculatively. If it is found that the branch was guessed incorrectly when the condition-code-setting instruction finishes, all instructions beyond the associated branch are canceled, and the correct instructions are then dispatched. The primary method for branch prediction for unresolved branches uses a branch-history table (BHT) containing 2048 prediction fields, each with a two-bit branch-history entry. The two-bit prediction field is a saturating up–down counter with 0 corresponding to strongly not-taken and 3 corresponding to strongly taken. When branches are resolved, the prediction field for that entry is incremented or decremented depending upon whether the branch was taken or not taken, respectively, except when the field is already saturated.

### Fixed-point execution units

The POWER3 processor contains three fixed-point execution units: two single-cycle units and one multi-cycle unit. The single cycle units execute all single-cycle instructions (arithmetic, shift, logical, compare, trap, and count leading zero) with single-cycle latency (this means that instructions dependent upon the result can execute in the next cycle). All other fixed-point instructions, such as multiply and divide, are handled by the multi-cycle unit. Since the POWER3 processor is a 64-bit microprocessor, this includes 64-bit as well as 32-bit integer operands. The two single-cycle fixed-point units share a six-entry instruction queue, while the multi-cycle unit includes a three-entry instruction queue. In contrast to the POWER2 processor, which included two symmetric units that executed both fixed point and load/store instructions, the POWER3 design includes two dedicated load/store units in addition to the three fixed-point units. The independence of the fixed point execution units and the load/store execution units is obviously a large performance benefit for calculations that are predominately integer in nature, such as Monte Carlo simulations. But even in floating-point calculations, this separation can be important. An example of this occurs in a sparse-matrix-vector multiply, which involves address indirection, whereby an integer index must be converted to byte-offset by a fixed-point instruction before it is used by a subsequent floating-point load operation.

### Floating-point execution units

The floating-point unit (FPU) contains two symmetrical floating-point execution units which implement a fused multiply–add pipeline conforming to the PowerPC Architecture. All floating-point instructions pass through both the multiply stage and the add stage. For floating point multiplies, 0 is used as the add operand, and for floating-point adds, 1 is used as the multiplicand. Each floating-point execution unit supports three cycle data forwarding for dependent instructions within the same execution unit when the target of the first instruction feeds either the FRB or the FRC operand of the dependent instruction, where the operation is FRT 4 [(FRA) 3 (FRC)] 1 (FRB). In the case of data forwarding between execution units, or when, on the same execution unit, the first instruction is feeding the FRA operand of the dependent instruction, the latency is four cycles. It is worth noting that, for achieving frequency targets, the pipeline of floating-point register-to-register instructions is broken up into ten stages (Fetch, Dispatch/Decode, Register Access, Execute 1, 2, 3, and 4, Finish, Complete, and Write back), but only the first three Execute stages are exposed for dependent instructions. Most floating-point instructions have single-cycle throughput. Since the POWER3 processor can execute two floating-point multiply–add instructions per cycle, the peak floating-point rate of the machine is four floating point operations per processor cycle. The floating-point arithmetic operations that are not pipelined are square root (fsqrt and fsqrts) and divide (fdiv and fdivs). These operations can use either of the execution units and are assisted by additional logic to handle their numerical algorithms. The POWER3 processor implements the optional PowerPC instructions fres (single-precision floating-point reciprocal estimate) and frsqrte (floating-point reciprocal square-root estimate). These are often useful for boosting performance in applications that do not need the full accuracy provided by the divide and square-root instructions (e.g., some graphic routines). These fast estimate instructions also provide the seed values for iterative divide and square-root routines. The SPPM example in this paper describes software vector versions of these routines, which perform significantly faster than the hardware divide and square-root instructions. The optional floating-point select instruction, fsel, is also implemented to provide for a floating-point conditional instruction with no branching. While this eliminates the chance of incurring the penalty for a mis-predicted branch, the more significant advantage in eliminating the branch is the increased flexibility provided to the compiler in scheduling a group of instructions that includes an fsel. The FPU also includes 32 64-bit floating-point registers and 24 64-bit physical rename registers or “buffers.” All target results of floating-point load and arithmetic instructions are placed in rename registers until the instruction completes (i.e., until the completion stage of the instruction). This method of using rename registers is vital to executing out of order, executing speculatively, and breaking false register dependencies. However, stalls may occur at some point if all rename registers are allocated. The POWER3 processor optimizes the use of its floating-point rename registers, which consume a large piece of premium silicon area on the chip. Typically, for microprocessors which implement register renaming, rename registers are allocated when instructions are dispatched. While the POWER3 processor does allocate from its pool of 32 “virtual” rename registers during the dispatch cycle, it delays allocation of the physical rename registers until the cycle for which they are needed, typically the execute or finish stages. This technique makes better use of the physical rename registers and prevents them from becoming the source of a performance bottleneck. The result is that the POWER3 processor is able to sustain near-peak performance on key application kernels such as rank-n update and matrix–matrix multiply, which presses the execution and completion rate of floating point instructions to the maximum. A central FPU instruction queue above the twin floating-point units can hold up to eight floating-point instructions, helping to maintain a steady flow of work for the FPU. The execution units can pull instructions from the queue in an out-of-order fashion, allowing logically later instructions whose operands are available to bypass other instructions which are waiting for operands. This flexibility provides a performance advantage when executing legacy code scheduled for other micro architectures, or for variable delays such as stalls resulting from cache misses. In addition to out-of-order issue, out-of-order finish capability allows faster but independent instructions to bypass slower ones. A common example is an instruction stream containing a floating-point divide followed by a series of FMAs which are independent of the divide; while one execution unit is executing the divide, the other instructions execute and finish in parallel in the other execution unit.

### Load/store execution units

All loading and storing of data is handled by two load/store execution units. Load instructions transfer data from memory to a fixed- or floating-point rename register; store instructions do just the opposite, transferring data from a register to memory. (Since the POWER3 processor is cache-based, data from a load may be found in the L1 or L2 cache, or a cache-line transfer from memory may be initiated as a result of the load.) The performance of store instructions is enhanced by the presence of a store buffer, which are 16 entries deep. Store instructions can finish executing if they have obtained their data; they do not have to wait until the data is written into the data cache. The POWER3 processor also implements load and store update form instructions, which update the general purpose register containing the load or store address as part of the instruction, eliminating the need for a separate add instruction. Using load update forms allows the compiler to generate concise code which maximizes the computational work performed within the four-instruction completion rate per cycle. A common example is a matrix vector multiplication, which primarily consists of two lfdu instructions and two fma instructions per cycle; this sequence will run at the peak flop rate of the processor when its data operands are contained in the L1 cache. The two load/store execution units share a six-entry instruction queue. The out-of-order LSU also permits load instructions to bypass store instructions while keeping track of any data dependencies that might exist, further enhancing performance and instruction scheduling flexibility. Order among store instructions is always maintained in both the execute stage and the store queue.

### Data cache unit

The data cache unit (DCU) consists primarily of the data memory management unit (MMU), the primary (L1) data cache, and the data prefetch unit.

### Memory management unit

The MMU is primarily responsible for address translation of data requests. It includes a 16-entry segment look aside buffer (SLB) and two mirrored 256-entry two-way set associative data translation look aside buffers (DTLB) in support of the dual load/store execution units. For translation misses, the instruction and data share hardware to search the page table for the correct translation. The MMU supports one terabyte (240 bytes) of physical memory, 64-bit effective addressing, and 80-bit virtual addressing.

### L1 data cache

The L1 data cache is 64 KB in size and is designed to provide single-cycle access to the load/store units. The data cache consists of 512 128-byte cache lines organized into four banks. Each bank is 128-way set-associative; that is, any line with the two address bits (A55/A56) set to select a particular bank may reside in any of the 128 cache slots in that bank. Within each bank, there are two sub-banks, one for all even double words and one for all odd double words (selected by address bit A60). The data cache can return the operands for two loads in the same cycle provided they are not both in the same bank and they are also not in the same even or odd double word sub-bank (i.e., if A55, A56, and A60 all match for two loads, a conflict exists). While 128-way set associativity provides performance advantages, it makes a least-recently-used (LRU) replacement algorithm impractical to implement. Hence, the POWER3 processor implements a round-robin replacement scheme for both the L1 instruction cache and the L1 data cache. As data is transferred between memory and the processor, buses of various widths and frequencies are used. To provide a place to reconstruct a cache line, and to accommodate differences in frequencies and bus widths, line width buffers are often inserted in the transfer paths. Between the L1 data cache and the BIU, four cache reload buffers (CRBs) are used to stage data into the L1 data cache (each data cache bank has a dedicated CRB). Outgoing data from the data cache is staged through four cache-store back buffers (CSBs), one per bank. Load hits to data in the CRB are satisfied directly, rather than delayed until the cache line is reloaded into the L1 cache array. The CSB provides a 64-byte interface to the BIU that also facilitates a highly parallel and efficient DCU. The data unit can support up to four outstanding cache misses, providing the basis for reducing the effective latency to memory. When a load misses the L1 data cache, the instruction is placed in a six-entry load-miss queue while the BIU fetches the data. Subsequent loads can continue to execute. If a subsequent load hits the cache, its data is returned immediately. If a subsequent load misses the cache, it is also placed in the load-miss queue. Since the BIU and memory subsystem can overlap the requests to memory for the missed cache lines, the average latency of memory per miss is reduced when there is more than one cache miss outstanding. Only after there are four outstanding misses in the load-miss queue and a fifth miss is encountered does load execution stall until one of the load instructions in the LMQ is serviced. Loads that hit the cache while there are four outstanding cache misses will continue to execute.

### Data pre-fetching

One of the most effective and innovative features of the POWER3 processor is its hardware data pre-fetch capability. The POWER3 processor pre-fetches data by monitoring data cache-line misses and detecting patterns. When a pattern, herein called a stream, is detected, the POWER3 processor speculatively pre-fetches cache lines in anticipation of their use. With memory latency improving at a slower pace than processor cycle time, data pre-fetching is extremely advantageous in hiding the memory latency in order to achieve adequate bandwidth for data-hungry applications. Pre-fetched streams have data storage patterns that reference consecutive cache lines, either in order of increasing addresses or decreasing addresses. It has been observed by the authors and by others that a high percentage of data reference patterns in engineering/scientific applications conform to this pattern. Because of the economies of cache-based processors, new and rewritten applications give preference to such consecutive data access patterns. Even many so-called sparse data structures store the bulk of data in a stride-1 fashion, while the indirect addressing associated with the sparsity is contained within a cacheable region. Cache-miss patterns that are random or at a stride (in cache lines) greater than one (e.g., every fourth line) will not cause hardware pre-fetches, and attempting to pre-fetch the latter case would greatly increase the complexity of the hardware; such patterns are already handled adequately by the multiple outstanding miss capability of the POWER3 processor (since each access would be a distinct cache line). The POWER3 processor pre-fetch engine includes a ten entry stream filter and a stream pre-fetcher (Figure 2). The purpose of the stream filter is to observe all data cache line misses, in the form of real addresses (RA in Figure 2) and to detect possible streams to pre-fetch. The stream filter records data cache-line miss information; the real address of the cache line is incremented or decremented (depending upon the offset within the line corresponding to load operand), and this “guess” is placed in the FIFO filter. As new cache misses occur, if the real address of a new cache miss matches one of the guessed addresses in the filter queue, a stream has been detected. If the stream pre-fetcher has fewer than four streams active, the stream is installed, and a pre-fetch to the line anticipated next in the stream is sent out via the BIU. Once placed in the stream pre-fetcher, a stream remains active until it is aged out. Normally a stream is aged out when the stream reaches its end and other cache misses displace its entry in the stream filter. When a stream is being pre-fetched, the pre-fetcher tries to stay two cache lines ahead of the current line (i.e., the line whose elements are currently being accessed by a load). The cache line that is one line ahead of the current line is pre-fetched into the L1 cache, and the line which is two ahead of the current line is pre-fetched into a special pre-fetch buffer in the BIU. Hence, the pre-fetch engine can concurrently pre-fetch four streams, each of which may be up to two lines ahead of the current line, for a total of eight pre-fetches per processor. The pre-fetch engine monitors all load addresses from the LSU (EA0 and EA1 in Figure 3). When the LSU finishes with the current line and advances to the next line (which is already in the L1 cache because of pre-fetching), the pre-fetch hardware transfers the line which is in the pre-fetch buffer to the L1 and pre-fetches the next line into the buffer. In this way, the pre-fetching of lines is automatically paced by the rate at which elements in the stream are consumed.

### Bus interface unit

The bus interface unit (BIU) provides the interface between the processor bus and the other processor units: the instruction processing unit, the data cache unit, the prefetch engine, and the L2 cache. Its memory bus interface is 128 bits (16 bytes) wide and supports a variety of processor-to-bus clock ratios.

### L2 cache

The POWER3 processor supports a private, either direct mapped or set-associative, unified instruction and data secondary (L2) cache with sizes from 1 MB to 16 MB. The private bus to the L2 is 32 bytes wide, and cache-line transfers of the supported 128-byte line are performed in a burst operation of four cycles. For the 375-MHz RS/6000 44P Model 270, which runs the L2 interface at a ratio of 3:2 with the processor, this produces a burst rate of 8 GB/s. The 43P Model 270 also has load-use latency (L1 miss, L2 hit instance) of approximately twelve cycles. The wide high-speed L2 cache interface provides ample bandwidth for processor requests as well as for snoop traffic from other processors or node controllers in the system.

Stream filter

Load miss queue

Stream prefetch control

Stream

allocation Control

Bus interface unit

Prefetch guess

logic

Stream prefetcher

RA

Figure (2) the prefetch engine.

# IBM Power4 System Micro Architecture

## Chip Features

* Each processor has private L1 caches
* Both processors share an on-chip L2 cache through a core interface unit (CIU)
* Crossbar between two processors’ I and D L1 caches and three L2 controllers
* Each L2 controller can feed 32B per cycle
* Accepts 8B processor stores to L2 controllers
* Each processor has a non-cacheable unit (NC)
* L3 directory and L3 controller on chip

## Processor Features

* Dual processor core
* 8-way superscalar
* Out of Order execution
* 2 Load / Store units
* 2 Fixed Point units
* 2 Floating Point units
* Logical operations on Condition Register
* Branch Execution unit
* 200 instructions in flight, Hardware instruction and data prefetch
* **L3** High-speed connections to up to 3 other pairs of POWER4 CPUs
* Ability to turn off pair of CPUs to increase throughput
* Apple G5 uses a single-core derivative of POWER4 (PowerPC 970)
* POWER4 systems were designed to handle up to 32 physical processors on 16 chips

## Performance enhancement techniques

### Pipeline Requirements

* Maintain binary compatibility
* Maintain structural compatibility
* Optimizations for POWER4 carry forward
* Improved performance
* Enhancements for server virtualization
* Improved reliability, availability, and serviceability at chip and system levels

### Pipeline Improvements

* Enhanced thread level parallelism
* Two threads per processor core
* a.k.a Simultaneous Multithreading (SMT)
* 2 threads/core \* 2 cores/chip = 4 threads/chip
* Each thread has independent access to L2 cache
* Dynamic Power Management
* Reliability, Availability, and Serviceability

### Load/Store Unit Operation

* Main structures
* Load Reorder Queue (LRQ), i.e., load buffer
* Store Reorder Queue (SRQ), i.e., store address buffer
* Store Data Queue (SDQ)
* Hazards avoided by Load/store unit

## Memory Hierarchy

### Cache Coherence

Each L2 controller has four coherency processors to handle requests from either processor’s caches or store queues

Multi Chip Module (MCM) Architecture

* 4 processor chips
* 2 processors per chip
* 8 off-module L3 chips
* L3 cache is controlled by MCM and logically shared across node
* 4 Memory control chips
* 16 chips

# IBM Power5 System Micro architecture

* The Power5 processor implements the 64-bit PowerPC\* architecture.
* Two identical processor cores are contained on a single die. The processor cores each support two logical threads
* This microprocessor allows system scalability to 64 physical Processors. And system allows both single-threaded and multithreaded execution modes
* The POWER5 microprocessor implements dynamic resource balancing to ensure that each thread receives its fair share of system resources.
* Which is designed to allow for POWER4 optimizations to carry over
* The POWER5\* chip is the next-generation chip. In designing the POWER5 system, a key goal was to maintain both binary and structural compatibility with existing POWER4 systems to ensure not only those binaries would continue to execute properly, but that all application optimizations would carry forward to newer systems
* High floating point performance
* Server flexibility
* Power efficient design
* Utility:Reliability,availability,servicesability
* Dynamic Power Management with no performance impact

****

Figure (3) Block diagram of the POWER5

## System feature

* Dual core processor
* Shared L2 cache
* Shared L3 cache
* Shared Memory
* Multiple Page Size support

Simultaneous Multi Threading

## Processor Characteristics

* Deep pipelines High frequency clocks
* High asymptotic rates
* Superscalar
* Speculative out-of-order instructions
* Up to 8 outstanding cache line misses
* Large number of instructions in flight
* Branch prediction
* Pre-fetching

## Micro partitioning

* Up to 64 physical processors,1280 virtual processors per system

### Multi Chip Module (MCM) Architecture

* 4 processor chips
* 2 processors per chip
* 4 L3 cache chips
* L3 cache is used by processor pair
* “Extension “of L2
* 8 chips

### Registers

* CPU's point of view
* 120 FP registers (POWER5)
* User point of view
* 32 FP registers (architecture)
* Rename registers
* Relieve register "pressure"

## Performance enhancement technique

### Chip Stats

* Copper interconnects
* Decrease wire resistance and reduce delays in wire dominated chip timing paths
* 8 levels of metal
* 389 mm2
* Silicon on Insulator devices (SOI)
  + Thin layer of silicon (50nm to 100 µm) on insulating substrate, usually sapphire or silicon dioxide (80nm)
  + Reduces electrical charge transistor has to move during switching operation (compared to CMOS)
    - Increased speed (up to 15%)
    - Reduced switching energy (up to 20%)
    - Allows for higher clock frequencies (> 5GHz)
  + SOI chips cost more to produce and are therefore used for high-end applications
  + Reduces soft errors

### Pipeline

* Pipeline identical to POWER4
* All latencies including branch misprediction penalty and load-to-use latency with L1 data cache hit same as POWER4

### Instruction Fetch

* Fetch up to 8 instructions per cycle from instruction cache
* Instruction cache and instruction translation shared between threads
* One thread fetching per cycle

### Branch Prediction

* Three branch history tables shared by 2 threads
* 1 bimodal, 1 path-correlated prediction
* 1 to predict which of the first 2 is correct
* Can predict all branches – even if every instruction fetc Branch to link register (bclr) and branch to count register targets predicted using return address stack and count cache mechanism
* Absolute and relative branch targets computed directly in branch scan function
* Branches entered in branch information queue (BIQ) and reallocated in program order hed is a branch

### Instruction Grouping

* Separate instruction buffers for each thread
* 24 instructions / buffer
* 5 instructions fetched from 1 thread’s buffer and form instruction group
* All instructions in a group decoded in parallel

### Performance Expectations

* Higher sustained-to-peak floating point rate ratio compared to POWER4
* Reduction in L3 and memory latency
* Integrated memory controller
* Increased rename resources
* Higher instruction level parallelism in compute intensive applications
* Fast barrier synchronization operation
* Enhanced data pre-fetch mechanism

# IBM POWER6 System Micro Architecture

## Introduction

The POWER6 is a [microprocessor](http://en.wikipedia.org/wiki/Microprocessor) developed by [IBM](http://en.wikipedia.org/wiki/IBM) that implemented the [Power ISA v.2.03](http://en.wikipedia.org/wiki/Power_Architecture#Power_ISA_v.2.03). When it became available in systems in 2007, it succeeded the [POWER5+](http://en.wikipedia.org/wiki/POWER5#POWER5.2B) as IBM's flagship Power microprocessor. It is part of the eCLipz project, said to have a goal of converging IBM's server hardware where practical (hence "ipz" in the acronym: [iSeries](http://en.wikipedia.org/wiki/ISeries), [pSeries](http://en.wikipedia.org/wiki/PSeries), and [zSeries](http://en.wikipedia.org/wiki/ZSeries)).

POWER6 was described at the [International Solid-State Circuits Conference](http://en.wikipedia.org/wiki/International_Solid-State_Circuits_Conference) (ISSCC) in February 2006, and additional details were added at the Microprocessor Forum in October 2006 and at the next ISSCC in February 2007. It was formally announced on May 21, 2007. It was released on June 8, 2007 at speeds of 3.5, 4.2 and 4.7 GHz, but the company has noted prototypes have reached 6 GHz. POWER6 reached first silicon in the middle of 2005, and was bumped to 5.0 GHz in May 2008 with the introduction P595.

## Description

The POWER6 is a [dual-core](http://en.wikipedia.org/wiki/Dual-core) processor. Each core is capable of two-way [simultaneous multithreading](http://en.wikipedia.org/wiki/Simultaneous_multithreading) (SMT). The POWER6 has approximately 790 million transistors and is 341 mm2 large fabricated on a [65 nm](http://en.wikipedia.org/wiki/65_nanometer) process. A notable difference from [POWER5](http://en.wikipedia.org/wiki/POWER5) is that the POWER6 executes instructions in-order instead of [out-of-order](http://en.wikipedia.org/wiki/Out-of-order_execution). This change often requires software to be recompiled for optimal performance, but the POWER6 still achieves significant performance improvements over the POWER5+ even with unmodified software, according to the lead engineer on the POWER6 project.

POWER6 also takes advantage of [ViVA-2](http://en.wikipedia.org/wiki/IBM_ViVA), Virtual Vector Architecture, which enables the combination of several POWER6 nodes to act as a single [vector processor](http://en.wikipedia.org/wiki/Vector_processor).

Each core has two [integer units](http://en.wikipedia.org/wiki/Arithmetic_logic_unit), two [binary](http://en.wikipedia.org/wiki/Binary_code) [floating-point units](http://en.wikipedia.org/wiki/Floating_point_unit), an [AltiVec](http://en.wikipedia.org/wiki/AltiVec) unit, and a novel [decimal](http://en.wikipedia.org/wiki/Decimal_floating_point) floating-point unit. The binary floating-point unit incorporates “many micro architectures, logic, circuit, latch and integration techniques to achieve [a] 6-cycle, 13-[FO4](http://en.wikipedia.org/wiki/FO4) pipeline,” according to a company paper. Unlike the servers from IBM's competitors, the POWER6 has hardware support for [IEEE 754](http://en.wikipedia.org/wiki/IEEE_754) decimal arithmetic and includes the first decimal [floating-point](http://en.wikipedia.org/wiki/Floating-point) unit integrated in silicon. More than 50 new floating point instructions handle the decimal math and conversions between [binary](http://en.wikipedia.org/wiki/Binary128) and [decimal](http://en.wikipedia.org/wiki/Decimal128). This feature was also added to the [z10](http://en.wikipedia.org/wiki/IBM_z10_%28microprocessor%29) microprocessor featured in the [System z10](http://en.wikipedia.org/wiki/IBM_System_z10).

Each core has a 64 KB, four-way set-associative instruction cache and a 64 KB data cache of an eight-way set-associative design with a two-stage pipeline supporting two independent 32-bit reads or one 64-bit write per cycle. Each core has semi-private 4 [MiB](http://en.wikipedia.org/wiki/MiB) unified [L2 cache](http://en.wikipedia.org/wiki/L2_cache), where the cache is assigned a specific core, but the other has a fast access to it. The two cores share a 32 MiB [L3 cache](http://en.wikipedia.org/wiki/L3_cache) which is off die, using an 80 GB/s bus.

POWER6 can connect to up to 31 other processors using two inter node links (50 GB/s), and supports up to 10 logical partitions per core (up to a limit of 254 per system). There is an interface to a service processor that monitors and adjusts performance and power according to set parameters.

IBM also makes use of a 5 GHz duty-cycle correction clock distribution network for the processor. In the network, the company implements a copper distribution wire that is 3 µm wide and 1.2 µm thick. The POWER6 design uses dual power supplies, a logic supply in the 0.8-to-1.2 Volt range and an SRAM power supply at about 150-mV higher.

The thermal characteristics of POWER6 are similar to that of the [POWER5](http://en.wikipedia.org/wiki/POWER5). [Dr Frank Soltis](http://en.wikipedia.org/wiki/Frank_Soltis), an IBM chief scientist, said IBM had solved power leakage problems associated with high frequency by using a combination of [90 nm](http://en.wikipedia.org/wiki/90_nanometer) and 65 nm parts in the POWER6 design.

IBM introduced POWER6\* microprocessor-based systems in 2007. Based upon the proven simultaneous multithreaded (SMT) implementation and dual-core technology in the POWER5\* chip, the design of the POWER6 microprocessor extends IBM leadership by introducing a high-frequency core design coupled with a cache hierarchy and memory subsystem specifically tuned for the ultrahigh-frequency multithreaded cores. The POWER6 processor implements the 64-bit IBM Power Architecture\* technology. Each POWER6 chip incorporates two ultrahigh-frequency dual threaded SMT processor cores, a private 4-MB level 2 cache (L2) for each processor, a 32-MB L3 cache controller shared by the two processors, two integrated memory controllers, an integrated I/O controller, an integrated symmetric multiprocessor (SMP) coherence and data interconnect switch, and support logic for dynamic power management, dynamic configuration and recovery, and system monitoring. The SMP switch enables scalable connectivity for up to 32 POWER6 chips for a 64-way SMP. The ultrahigh-frequency core represents a significant change from prior designs. Driven by the latency and throughput requirements of the new core, the large, private L2 caches represent a departure from the designs of the POWER4\* and POWER5 processors, which employed a smaller, shared L2 cache. The large, victim L3 cache, shared by both cores on the chip and accessed in parallel with the L2 caches, is similar in principle to the POWER5 L3 cache, despite differences in the underlying implementation resulting from the private L2 caches. Likewise, the integrated memory and I/O controllers are similar in principle to their POWER5 counterparts. The SMP interconnect fabric and associated logical system topology represent broad changes brought on by the need to enable improved reliability, availability, and serviceability (RAS), virtualization, and dynamic configuration capabilities. The enhanced coherence protocol facilitates robust scalability while enabling improved system packaging economics. In this paper, we focus on the micro architecture and its impact on performance, power, system organization, and cost. We begin with an overview of the key features of the POWER6 chip, followed by detailed descriptions of the ultrahigh-frequency core, the cache hierarchy, the memory and I/O subsystems, the SMP interconnect, and the advanced data pre-fetch capability. Next, we describe how the POWER6 chipset can be employed in diverse system organizations.

## High-frequency core design

The POWER6 core is a high-frequency design that is optimized for performance for the server market as well as power. It provides additional enterprise functions and RAS characteristics that approach mainframe offerings.



Figure (4) Evolution of the POWER6 chip structure (SMT2: a dual-threaded simultaneous multithread.)

Its 13-FO41 pipeline structure yields a core whose frequency is two times that of the 23-FO4 POWER5 core. The function in each pipeline stage is tuned to minimize excessive circuitry, which causes delay and consumes excessive power. Speculation, which is costly at high frequency, is minimized to prevent wasted power dissipation. As a result, register renaming and massive out-of-order execution as implemented in the POWER4 and POWER5 processor designs are not employed. The internal core pipeline, which begins with instruction fetching from the instruction cache (I-cache) through instruction dispatch and execution, is kept as short as possible. The instruction decode function, which consumed three pipe stages in the POWER5 processor design, is moved to the pre-decode stages before instructions are written into the I-cache. Delay stages are added to reduce the latency between dependent instructions. Execution latencies are kept as low as possible while cache capacities and associativity are increased. The POWER6 core has twice the cache capacity of its predecessor, providing one-cycle back-to-back fixed-point (FX) execution on dependent instructions, a two-cycle load for FX instructions, and a six-cycle floating-point (FP) execution pipe. The number of pipeline stages of the POWER6 processor design (from instruction fetch to an execution that produces a result) is similar to the POWER5 processor stages, yet the POWER6 core operates at twice the frequency of the POWER5 core. In place of speculative out-of-order execution that requires costly circuit renaming, the POWER6 processor design concentrates on providing data pre-fetch. Limited out-of-order execution is implemented for FP instructions. Dispatch and completion bandwidth for SMT has been improved. The POWER6 core can dispatch and complete up to seven instructions from both threads simultaneously. The bandwidth improvement, the increased cache capacity, cache associatively, and other innovations allow the POWER6 core to deliver better SMT speedup than the POWER5 processor-based system. Power management was implemented throughout the core, allowing a clock gating efficiency2 of better than 50%.

## Balanced system throughput

While the frequency trade-offs were appropriate for the core, it did not make sense to extend ultrahigh frequency to the cache hierarchy, SMP interconnect, memory subsystem, and I/O subsystem. In the POWER5 processor design, the L2 cache operates at core frequency, and the remaining components at half that frequency. Preserving this ratio with the higher relative frequency of the POWER6 core would not improve performance but would actually impair it, since many latency penalties outside the core are more tied to wire distance than device speeds. Because the latencies in absolute time tend to remain constant, incorporating a higher-frequency clock results in added pipeline stages. Given that some time is lost every cycle because of clocking overhead, the net effect is to increase total latency in absolute time while increasing the power dissipation due to the increase in pipeline stages. Therefore, for the POWER6 processor design, the L2 cache, SMP interconnect, and parts of the memory and I/O subsystems operate at half the core frequency, while the L3 cache operates at one-quarter, and part of the memory controller operates at up to 3.2 GHz. With lower power and slower devices, chip power is reduced. Because of their lower speed relative to the core, these components must overcome latency and bandwidth challenges to meet the balanced system performance requirements. To achieve a balanced system design, all major subsystems must realize similar throughput improvements, not merely the cores. The cache hierarchy, SMP interconnects fabric, memory subsystem, and I/O subsystem must keep up with the demands for data generated by the more-powerful cores.



Table (1) POWER5 to POWER6 processor throughput comparison (relative to core cycles)

Therefore, for the POWER6 processor design, the internal data throughput was increased commensurately with the increase in processing power, as shown in Table 1.

Since the L2 cache was designed to operate at half the frequency of the core, the width of the load and store interfaces was doubled; instead of driving 32 bytes of data per core cycle into the core, the POWER6 processor L2 drives an aggregate of 64 bytes every other core cycle. While the POWER5 processor L2 can obtain higher peak bandwidth when simultaneously delivering data to the data cache (D-cache) and I-cache, in realistic situations the D-cache and I-cache do not drive high-throughput requirements concurrently. This is because high bus utilization due to D-cache misses typically occurs in highly tuned single-threaded scenarios when there are multiple outstanding load instructions continuously in the pipeline, while I-cache misses interrupt the flow of instructions into the pipeline. Instead of accepting 8 bytes of store data per core cycle, the POWER6 processor L2 accepts 16 bytes of store data every other core cycle. Note that the aggregate bandwidth of the POWER6 processor L2 per core per cycle is two thirds that of the POWER5 processor L2. It does not have to scale perfectly for the following reasons: The POWER6 core has larger L1 caches, so there are fewer L1 misses driving fetch traffic to the L2; the POWER6 processor L2 can manage store traffic with 32-byte granularity, as opposed to 64-byte granularity for the POWER5 processor L2, so normally there is less L2 bandwidth expended per store. In addition, the POWER6 processor L2 is much larger per core than the POWER5 processor L2, so there are fewer L2 misses, driving fewer cast out reads and allocate writes. (The term cast out refers to the movement of deallocated, modified data from a given level in the cache hierarchy either to the next level of cache or to memory.)

For the POWER6 chip, the IBM Elastic Interface (EI) logic, which is used to connect to off-chip L3 cache data chips, I/O bridge chips, and SMP connections to other POWER6 chips, was accelerated to operate at one-half of the core frequency, keeping pace with corresponding interfaces in prior designs by achieving significantly higher frequency targets. The POWER6 processor L3 cache can read up to 16 bytes and simultaneously write up to 16 bytes every other core cycle, just as the POWER5 processor L3. The POWER6 processor off-chip SMP interconnect comprises five sets of links. The organization of these is described later in the section ‘‘SMP interconnect.’’ Each set can import up to 8 bytes and simultaneously export up to 8 bytes of data or coherence information every other core cycle. While this does not match the POWER5 processor SMP interconnect bandwidth per core cycle as seen by a given chip, the difference in system topology (described later in the section ‘‘SMP interconnect’’) and an increased focus on hypervisor and operating system optimizations for scalability drive a relaxation for the demand for interconnect data bandwidth. The EI logic used for connectivity to off-chip memory buffer chips was accelerated to operate at 3.2 GHz when interacting with 800-MHz DRAM (dynamic random access memory) technology. By using both integrated memory controllers, a single POWER6 chip can read up to 16 bytes of data and simultaneously write up to 8 bytes of data or commands at 3.2 GHz. The I/O controller can read up to 4 bytes and simultaneously write up to 4 bytes of data to an I/O bridge chip every other core cycle.

## Processor core

The POWER6 core micro-architecture was developed to minimize logic content in a pipeline stage. Circuit area and speculative work are minimized in order to reduce wasted power dissipation. The result is a 13-FO4 design with short pipeline, large split L1 instruction, and D-caches supporting two-way SMT. Additional virtualization functions, decimal arithmetic, and vector multimedia arithmetic were added. Checkpoint retry and processor sparing were implemented. Instruction fetching and branch handling are performed in the instruction fetch pipe (Figure 2). Instructions from the L2 cache are decoded in pre-code stages P1 through P4 before they are written into the L1 I-cache.



**Figure 3**

Figure (5) POWER6 core pipeline (AG: address generation; BHT: branch table access and predict; BR: branch; DC: data-cache access; DISP: dispatch; ECC: error-correction code; EX: execute; FMT: formatting; IB: instruction buffer; IC0/IC1: instruction-cache access; IFA: instruction fetch address; ISS: issue; P1–P4: pre-decode; PD: post-decode; RF: register file access.)

Branch prediction is performed using a 16K-entry branch history table (BHT) that contains 2 bits to indicate the direction of the branch. Up to eight instructions are then fetched from the L1 I-cache and sent through the instruction decode pipeline, which contains a 64-entry instruction buffer (I-buffer) for each thread.

Instructions from both threads are merged into a dispatch group of up to seven instructions and sent to the appropriate execution units. Branch and logical condition instructions are executed

in the branch and conditional pipeline, FX instructions are executed in the FX pipeline, load/store instructions are executed in the load pipeline, FP instructions are executed in the FP pipeline, and decimal and vector multimedia extension instructions are issued through the FP issue queue (FPQ) and are executed in the decimal and vector multimedia execution unit. Data generated by the execution units is staged through the checkpoint recovery (CR) pipeline and saved in an error-correction code (ECC)-protected buffer for recovery. The FX unit (FXU) is designed to execute dependent instructions back to back.

The FP execution pipe is six stages deep. Figure 4 shows the pipeline for both the POWER6 and the POWER5 design with cycle-time delay for each stage, starting with instruction fetching from the I-cache to the time the result is available for subsequent instruction. FX load/store instructions are executed in order with respect to each other, while FP instructions are decoupled from the rest of the other instructions and allowed to execute while overlapping with subsequent load and FX instructions. Additional emphasis was put in the design to minimize the memory effect: pre-fetching to multiple levels of caches, speculative execution of instructions to pre-fetch data into the L1 D-cache, speculative pre-fetching of instructions into the L1 I-cache, providing a load-data buffer for FP load instructions, hardware stride pre-fetching, and software-directed pre-fetching. Buffered stages are added between the dispatch stage and the execution stage to optimize certain execution latency between categories of instructions. The following are examples:

* The FX instructions are staged for two additional cycles prior to execution in order to achieve a one- cycle load-to-use between a load instruction and a dependent FX instruction.



Figure (6) Internal POWER6 processor pipeline compared with the POWER5 processor pipeline with cycle time.

* Branch instructions are staged for two additional cycles to line up with FX staging instructions in order to avoid an additional branch penalty on an incorrect guess.
* FP instructions are staged through the FPQ for six cycles to achieve a zero-cycle load-to-use instruction between a load instruction and the dependent FP instruction.

The POWER6 core contains several units. The instruction fetch unit (IFU) performs instruction fetching, instruction pre-decoding, branch prediction, and branch execution. The instruction dispatch unit (IDU) performs instruction dispatch, issuing, and interrupt handling. The two FXUs, the two binary FP units (BFUs), the decimal FP unit (DFU), and the vector multimedia extension (VMX) unit are responsible for executing the corresponding set of FX, FP, decimal, and VMX instructions. The two load/store units (LSUs) perform data fetching. The recovery unit (RU) contains the data representing the state of the processor that is protected by ECC so that the state of the processor can be restored when an error condition is detected.

## POWER6+

The slightly enhanced POWER6+ was introduced in April 2009, but had been shipping in [Power 560 and 570](http://en.wikipedia.org/wiki/IBM_Power_Systems) systems since October 2008. It added more memory keys for secure memory partition, a feature taken from IBM's [mainframe processors](http://en.wikipedia.org/wiki/Z/Architecture).

## Products

As of 2008[[update]](http://en.wikipedia.org/w/index.php?title=POWER6&action=edit), the range of POWER6 systems includes "Express" models (the 520, 550 and 560) and Enterprise models (the 570 and 595). The various system models are designed to serve any sized business. For example, the 520 Express is marketed to small businesses while the Power 595 is marketed for large, multi-environment data centers. The main difference between the Express and Enterprise models is that the latter include Capacity Upgrade on Demand (CUoD) capabilities and hot-pluggable processor and memory "books". All Power systems are noted for their excellent scalability and storage capabilities.

|  |  |  |  |
| --- | --- | --- | --- |
| IBM POWER6 servers | | | |
| **Name** | **Number of sockets** | **Number of cores** | **CPU clock frequency** |
| 520 Express | 2 | 4 | 4.2 GHz or 4.7 GHz |
| 550 Express | 4 | 8 | 4.2 GHz or 5.0 GHz |
| 560 Express | 8 | 16 | 3.6 GHz |
| 570 | 8 | 16 | 4.4 GHz or 5.0 GHz |
| 570 | 16 | 32 | 4.2 GHz |
| 575 | 16 | 32 | 4.7 GHz |
| 595 | 32 | 64 | 4.2 GHz or 5.0 GHz |

Table (2)

IBM also offers four POWER6 based [blade servers](http://en.wikipedia.org/wiki/Blade_server). Specifications are shown in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| IBM POWER6 blade servers | | | |
| **Name** | **Number of cores** | **CPU clock frequency** | **Blade slots required** |
| BladeCenter JS12 | 2 | 3.8 GHz | 1 |
| BladeCenter JS22 | 4 | 4.0 GHz | 1 |
| BladeCenter JS23 | 4 | 4.2 GHz | 1 |
| BladeCenter JS43 | 8 | 4.2 GHz | 2 |

Table (3)

All blades support AIX, i, and Linux. The Blade Center S and H chassis is supported for blades running AIX, i, and Linux. The Blade Center E, HT, and T chassis support blades running AIX and Linux but not i.

At the Supercomputing 2007 (SC07) conference in Reno a new water-cooled Power 575 was revealed. The 575 is composed of 2U "nodes" each with 32 POWER6 cores at 4.7 GHz with up to 256 GB of RAM. Up to 448 cores can be installed in a single frame.

|  |  |  |  |
| --- | --- | --- | --- |
| IBM POWER6 disk storage | | | |
| **Name** | **Number of cores** | **CPU clock frequency** | **Number of controllers** |
| [DS8700](http://en.wikipedia.org/wiki/IBM_System_Storage#DS8000_Series) | 2, 4 | 4.7 GHz | 1, 2 |
| [DS8800](http://en.wikipedia.org/wiki/IBM_System_Storage#DS8000_Series) | 2, 4, 8 | 5.0 GHz | 1, 2 |

Table (4)

## Summary

With its high-frequency core architecture, enhanced SMT capabilities, balanced system throughput, and scalability extensions, the POWER6 microprocessor provides higher levels of performance than the predecessor POWER5 microprocessor-based systems while offering greater flexibility in system packaging trade-offs. Additionally, improvements in functionality, RAS, and power management have resulted in valuable new characteristics of POWER6 processor-based systems.

# IBM PowerPC 620 System Micro architecture

## Abstract

The PowerPC 620 superscalar microprocessor is the most recent and performance leading member of the PowerPC family, which is being jointly developed by IBM and Motorola. The 64-bit 620 represents the most aggressive micro architecture for superscalar processors to date. It employs a two-level branch prediction scheme, dynamic renaming for all the register files, distributed multi-entry reservation stations, true out-of-order execution by six pipelined execution units, and a completion buffer for ensuring precise interrupts.

A performance simulator for the 620 is developed using the VMW (Visualization-based Micro architecture Workbench) remerge table framework. The VMW-based simulator accurately models of a 620 micro architecture down to the machine cycle level. Extensive trace-driven simulation is performed using the SPEC9b2 benchmarks he experimental results indicate that the 620 is a well balanced design and achieves a maximum IPC rating of 1.94 on one of the benchmarks.

Detailed quantitative analyses of the effectiveness of all the key micro architecture features are presented. A brief philosophical comparison with the Alpha AXP 21164 is also including.

## Introduction

The latest announcement by the IBM-Motorola-Apple alliance is the PowerPC 620, the first 64-bit member and performance leader of the PowerPC family. While ~he latest Alpha AXP 21164 {8] (@ 300MHz) from DEC may edge out the 620 (@ 150MHz) as the industry performance leader~ the 620 employs the most aggressive micro architecture and achieves the highest level of instruction-level parallelism of any microprocessor currently on the market. The 620 is the first 64-hit superscalar microprocessor to employ true out-of-order execution, aggressive branch prediction, distributed multi-entry reservation stations, dynamic renaming for all register files~ six pipelined execution units, and a completion buffer to ensure precise interrupts. Most of these features have not been previously implemented in a single-chip microprocessor.

Their actual effectiveness is of great interest to both academic researchers as well as industry designers. This paper presents an instruction-level, or machine-cycle level, performance evaluation of the PowerPC 620 micro architecture.

## The PowerPC 620 Architecture

The PowerPC architecture is the result of the PowerPC alliance among Apple, IBM, and Motorola. It is based on IBM’s RS/6000 POWER architecture, designed to facilitate parallel instruction execution and be able to scale well with advancing technology. Motorola and IBM are the chief designers of the PowerPC architecture and the family of PowerPC chips, while Apple is focusing on PowerPC systems and software.

As of this writing, the PowerPC alliance has released three chips and announced a fourth. The first, which provided a transition from the POWER architecture to *PowerPC,* is the PowerPC601 [101. The second, low-power chip is the PowerPC60 3. Recently, a more advanced chip for desktop systems has begun production, the PowerP6C0 4. The fourth chip, the last one that will be designed by the alliance, is a high-performance chip. This chip, the PowerPC62 0 , was only recently announced.

The PowerPC architecture contains 32 integer registers (GPRs) and 32 floating point registers (FPRs). Also contains 32 condition register bits which can be addressed as one 32-bit register (CR), as a register with 8 four-bit fields (CRFs), or as 32 single-bit fields (CRBs). It also contains a count register (CTR) a link register (LR), both primarily used for branch instructions, and an integer exception register (XER) and a floating point status and control register (FPSCR) which s used to control the operation and record the exception status of the appropriate instruction types. PowerPC instructions are typical RISC, with the addition of floating point multiply-add fused (FMA) instructions and instructions to set, manipulate, and branch off of the condition register bits.

The most dominant architecture today, in terms of installed base, is the Intel x86. However the PowerPC appears to be making a serious challenge to this dominance. IBM is looking to the PowerPC as the new ISA for the entire company. Motorola has an agreement to supply Ford with PowerPC-based processors for future automotive electronics. Motorola is also planning on employing PowerPC processors in their products ranging from hand-held devices to multiprocessor servers. Future systems from Apple will employ PowerPC processors. In addition, Apple, IBM and Motorola have recently announced their agreement on developing common/compatible hardware platforms based on the PowerPC processors.

The 620 is a 4-wide superscalar machine. It uses aggressive branch prediction to fetch instructions as soon as possible and a generalized dispatch scheme to get those instructions to the execution units.

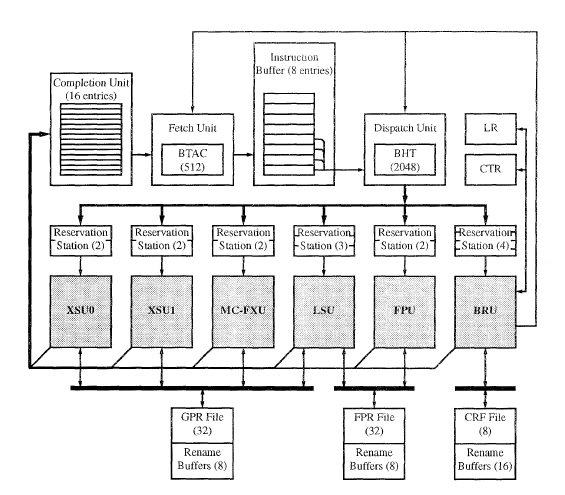


Figure (6) Power PC micro architecture diagram

The PowerPC 620 uses six parallel execution units. It has two simple (single-cycle) integer units, one complex (multi-cycle) integer unit, one floating-point unit (3 stages), one load/store unit (2 stages), branch unit. The branch unit accepts condition register logical instructions as well as branches. To keep these execution units as full as possible, the 620 uses distributed reservation stations and register renaming to implement an aggressive out-of-order execution scheme.

The 620 processes instructions are five major stages, some of which are separated by buffers to take up slack in dynamic variations of available parallelism. Major stages can require multiple cycles, while minor stages are one cycle each. The pipeline stages are **Fetch, Dispatch, Execute, Complete,** and **Write back.** The first three stages are followed by the Instruction Buffer, Reservation Stations, and the Completion Buffer, respectively. See figure 2.



Figure (6) power PC instruction Pipeline

**Fetch Stage:** The fetch unit accesses the instruction cache to retrieve up to four instructions per cycle into the instruction buffer. The end of a cache line or a taken branch can prevent the fetch unit from getting four useful instructions in a cycle, and a miss predicted branch can waste fetch cycles while fetching down the wrong path. Also during the fetch stage a preliminary branch prediction is made via the Branch Target Address Cache (BTAC) and the predicted next address is used for fetching in the next cycle.

**Instruction:** Buffer: The instruction buffer holds instructions between the fetch and dispatch stages. If the dispatch unit cannot keep up with the fetch unit, instructions will stay here for multiple cycles until the dispatch unit can get to them. A maximum of eight instructions can be buffered at a time. Instructions are buffered and shifted in groups of two to simplify the logic.

Dispatch Stage: The dispatch unit decodes instructions in the instruction buffer and checks if they can be dispatched to reservation stations. If all dispatch conditions are fulfilled for an instruction, the dispatch stage will allocate a reservation station entry, a completion buffer entry, and any needed destination rename registers for it. If these resources are available and each instruction goes to a different execution unit and all special serialization constraints are met, up to four instructions may be dispatched in order per cycle.

There are eight general-purpose register rename buffers, eight floating-point register rename buffers, and sixteen condition register field rename buffers. During dispatch, the necessary number of these buffers is allocated for the results of the instruction. Also during dispatch, any source operands which have been renamed by previous instructions are marked with the tags of the associated rename buffers. The reservation stations will then watch the appropriate result buses for forwarded results.

If a branch is being dispatched, resolution of the branch is attempted immediately. If the branch depends on an operand that is not yet available, it is predicted via the Branch History Table (BHT). If the predicted or resolved address does no match that made by the BTAC during the fetch stage, the speculatively fetched instructions are canceled and fetching restarts at the new address.

**Reservation Stations:** Each execution unit contains a reservation station to hold those instructions waiting to execute there, each reservation station can hold two to four instruction entries, depending on the execution unit. Each dispatched instruction waits in a reservation station until its entire source operands have been read or forwarded and the function unit is ready for execution. An instruction may then leave the reservation station and enter its function unit out of order. Each execution unit contains one reservation station and one reservation unit.

**Execute Stage**: A function unit computes the results of an instruction. This major stage can require multiple minors stages (cycles) to produce its results, depending on the type of instruction being executed. At the end of execution, the instruction results are sent to the destination rename buffers and forwarded to any waiting instructions, and the instruction is marked finished in the completion buffer.

**Completion Buffer**: The completion buffer holds the states of the in-flight instructions until they are architecturally complete. The completion buffer has sixteen entries with which to hold instruction information. An entry is allocated for each instruction during the dispatch stage. The execute stage then marks it as finished when the unit is done executing the instruction. Once an instruction is finished, it is eligible for completion.

**Complete Stage:** During the completion stage, finished instructions are removed from the completion buffer in order, up to four at a time, and passed to the write back stage. Fewer instructions will complete in a cycle if there are an insufficient number of write ports or if fewer than four instructions are finished and ready to complete in order. By holding instructions in the completion buffer until write back, the 620 guarantees that the architected registers hold the correct state up to the most recently completed instruction, thus allowing precise interrupts even with aggressive out-of-order techniques.

**Write back Stage:** During this stage, the write back logic moves the results of those instructions co completed by the completion unit in the last cycle from the rename buffers to the architected register files.

## Machine Specification

VMW is configurable to a specific machine implementation through four template tiles and a fifth C++ behavior tile. Two of the template files specify the syntax and semantics of the architected instructions, while the other two specify the organization and timing of the micro architecture. The C++ behavior file uses the specifications in the template files along with special-case routines that cannot be coded in the template files to model the execution of each instruction through the machine. To compare the micro architectural complexity of the PowerPC 620 with that of various other machines, we show a comparison with other simulators targeted using VMW.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Microprocessor** | **Alpha 21064** | **RS/6000** | **PowerPC 601** | **PowerPC 620** |
| Instruction Syntax (Templates) | 487 | 405 | 383 | 383 |
| Instruction Semantics (Templates) | 37 | 125 | 205 | 205 |
| Instruction Timing (Templates) | 24 | 87 | 112 | 113 |
| Machine Organization (Templates) | 39 | 60 | 51 | 75 |
| Machine Behavior (Lines of Code) | 668 | 1132 | 1080 | 3,093 |

Table (5) Comparison of Sizes of Machine Description Files

The five machine description files for the PowerPC 620 are generated based on internal confidential design documents provided and periodically updated by the PowerPC 620 design team. Correct interpretation of the design documents is checked by a member of the design team through a series of refinement cycles as the 620 design is finalized. Through this process, even late changes to the 620 design were incorporated into our machine description files and modeled by our performance simulator. Validation of the simulation model includes syntactic checking of the description files using tools provided by VMW and timing checking by comparing timing results on small benchmarks with lower-level hardware models.

## Experimental Framework

To analyze the PowerPC6 20 micro architecture, we used the Visualization-based Micro architecture Workbench (VMW), a retarget table performance simulator. The targeting of VMW for a specific processor is performed by specifying a set of machine description files. Once the files are specified, VMW provides the generic simulation and visualization engines and automatically compiles the files into a performance simulator for the specified machine.

## PowerPC 620 Instruction Fetching

Provided that there is enough room in the instruction buffer, the 620’s fetch unit is capable of fetching four instructions every cycle. That is, up until it encounters a branch. If the fetch unit were to wait for branch resolution before continuing to fetch non-speculatively, or if it were to bias naively fear branch not-taken., machine execution~ would be drastically slowed by the bottle neck in fetching down taken branches. Because of this, accurate branch prediction in a superscalar machine as wide as the PowerPC6 20 is extremely important. To demonstrate the branch prediction and speculation characteristics of the 620, we chose one of the integer benchmarks as an example case. We choose *espresso* because it has the most interesting characteristics.

## PowerPC 620 Instruction Dispatching

The PowerPC 620 dispatches instructions in order. This causes the dispatching unit to be a potential bottleneck. To begin with, however, the dispatch unit must be suitably supplied with instructions by the fetch unit. The rest of the examples in Sections 5 through 7 are from the *Alvin* benchmark in order to display the use of the floating-point functional units.

## Improved Out-of-Order Execution

Although other RISC vendors are just beginning to implement out-of-order execution, the 620 is the third PowerPC chip to use this technique. Like its predecessors, the new design includes “reservation stations” in each of the function units to hold instructions that are waiting for operands. The integer and floating-point units each have two reservations stations, as in the 604; the new branch unit has four reservation stations, permitting speculative execution beyond as many as four conditional branches. The 620 improves up on its predecessor in the load/store unit by adding a third reservation station.

The load/store unit will also perform some loads speculatively, that is, when a preceding branch instruction has been predicted but not resolved. To prevent operations that cannot be undone, the 620 will not speculatively execute stores of any kind or loads from non catchable (I/O) segments. Because most loads and stores are queued, as in the 604, these operations typically will not stall the processor pipeline.

The 620 handles unaligned loads and stores in hardware in both big- and little-endian modes, allowing it to do well in mixed environments with older Apple Macintoshes and x86-based PCs.

The other function units are quite similar to those in the 604. The integer units, of course, are expanded to 64 bits in width. The latency of floating-point loads is reduced from three cycles to two, and the latency of floating- point divides is improved to 18 cycles. The 620 adds a hardware square-root function that completes in 22 cycles; in the 604, square roots are calculated in software. These changes will improve performance on Spice circuit models, among other applications. As in the 604, all FP operations are performed in double-precision; there is no speed improvement for single-precision calculations.

## Two-Level Cache Structure Added

Although the 60x PowerPC chips can operate with second-level caches, all control functions for the L2 cache must be implemented in external logic. Furthermore, these chips share a single interface between the L2 cache and other system accesses. The 620 solves both of these problems, significantly improving performance.

## Memory Bandwidth Increased

The 620 system bus permits a low-cost 64-bit interface or a high-performance 128-bit design. Even the 64- bit version will provide much better memory bandwidth than the 64-bit 60x bus, as all cache traffic is shifted to the dedicated cache bus. This mode should be adequate for uni-processor servers or multiprocessor workstations.

The 128-bit mode may be required to meet the high bandwidth requirements of MP servers.

## Cost of Complexity Is High

The 620 uses a slightly different manufacturing process than the 603 and 604: a four-layer-metal version of a process that IBM calls CMOS-5S. This process reduces the gate oxide thickness, speeding the transistors. The faster transistors, along with improved circuit designs, deliver a clock rate 33% faster than that of the 604. The added features increase the die size of the 620 to 311 mm2, 59% larger than the 604. The new chip would have been even larger, but Motorola agreed to adopt IBM’s C4 (flip-chip) bonding method.

## Summary

The PowerPC6 20 does very well at branch prediction, often having zero delay cycles even for a taken branch. And even though precise interrupt is implemented, there is still a high degree of parallelism and out-of-order execution, especially with a well mixed stream of instructions, such as that in the *alvinn* benchmark. If the instruction mix is bustier, it tends to create a bottleneck in the dispatch unit. Even so, the integer benchmarks do quite well, and even the *hydro2d* benchmark it its high number of floating point divide instructions, still manages to achieve an IPC that RISC designers of a few years ago could only dream about.

One hot spot is the load/store unit. Although the difficulties of designing a two load/store unit system are myriad, the load/store bottleneck in the 620 is evident, and future, wider, processors will need that second unit. Having only one floating-point unit for four integer units is also a source of bottlenecks. The integer benchmarks rarely stall on the integer units, but the floating-point benchmarks stick waiting for FP resources often. The single dispatch to each reservation station in a cycle is also a serious source of dispatch stalls which can reduce the number of instructions considered for out-of-order execution.

# IBM PowerPC 970 System Micro Architecture

## Characteristics of PowerPC 970

PowerPC 970 micro architecture is designed for compute and bandwidth intensive applications. And the following capabilities

* Proven 64-bit architecture with native 32-bit PowerPC compatibility.
* AltiVec® Vector/SIMD engine for applications specific acceleration
* High bandwidth processor bus

The latest series of the PowerPC is introduced in 2002. The PowerPC 970 has significant deeper pipeline compare to the rest of its family. It is interesting to note that the architecture has a different number of pipelines depending on the type of instructions. There are 16 stages of Integer, 21 for floating point and 25 for SIMD operations. The variable pipeline stages objective is to achieve a consistent throughput for all the varying computational loads for different instructions. The 970 inherits the POWER4 wide superscalar execution unit. It has:

* 2 Float Point Units (FPUs)
* 2 Single Instruction Multiple Data (SIMDs)
* 2 Load/Store Unit (LSUs)
* A Branch and
* A condition-register unit

Although it has 8 EUs, it is still technically regarded as 5 ways superscalar, since the instruction retired rate is 5. The execution units are designed with additional resources (e.g. 2 parallel ALU units) to ensure a smooth flow of instructions throughout the entire pipeline, reducing control hazards that lead to stalls. The number of reservation station and stages for each EU are yet to be published. However one could estimate the number based on the POWER4 architecture.

We can see that architecture designers are exploiting the necessary hardware resource that is required by the application in the market. The AltiVec was first introduced in Motorola’s G4 processor and IBM later put this in the PPC 970 processor to meet the increasing computation demands of multi-media applications. It is therefore emphasizes that additional of multi-media instruction set to be included into general purpose computers. The AltiVec vector-processing unit has 128bit wide data paths and 32 registers of the same width to support multi-media operations.

Another special characteristic about the 970 is the grouping of internal operations. The 970 breaks instructions into internal operations (IOPs) and these IOPs are packaged into a group of 5 before dispatching into the execution core. The IOPs are grouped together for tracking purpose for the completion of in-order write back process. Instead of tracking individual IOPs, the completion unit tracks the IOP grouping, hence reducing the overhead associated with the tracking and reordering of instruction in the deep and wide pipeline.

Although the PowerPC 970 is a part that would cost considerably less to manufacture and sell, its performance actually exceeds the POWER4 processor in many areas. The reason for this apparent paradox is that the POWER4 processor had been designed for the high cost, continuous availability server market, and in some areas, performance had been traded off to obtain near-absolute reliability guarantees.

In figure 1, we show rough block diagrams of the PowerPC 970 processor and compare it against the POWER4 processor. Figure 1 show that one CPU core from the dual CPU core POWER4 processor is removed, the size of the L2 cache was resized to 512KB and optimized to connect to the single CPU core. A Vector SIMD unit was also added to the processor to handle the vector SIMD instruction extensions, and a new bus interface unit has been implemented to optimize for low CPU-count system performance.

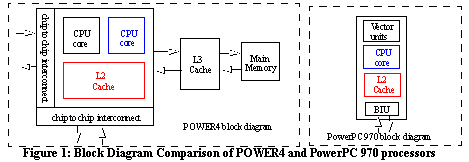


Figure (7) block diagram comparison of POWER4 and PowerPC 970 processors

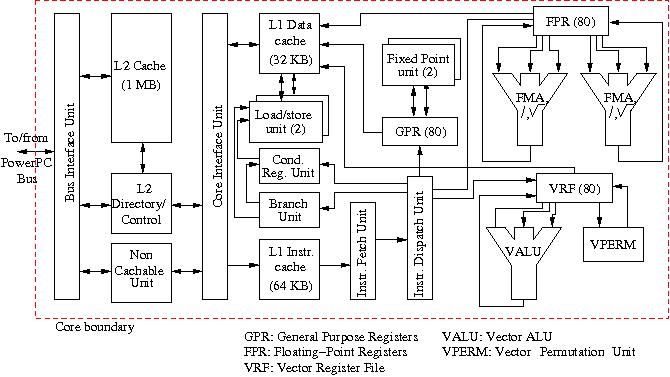


Figure (6) detail PowerPC 970 Micro Architecture

## Performance Enhancement Technique

### Altivec - Velocity Engine - Vector SIMD

In the PowerPC 970 processor, an Altivec-Compatible SIMD vector unit was added to the processor so that the processor can execute vector instructions from the Altivec vector SIMD ISA extension. Prior to the announcement of the PowerPC 970 processor, much speculation had been engaged as to the reason for the cryptic description of the "162 specialized SIMD instruction". It was well known that Altivec also had a similar number of SIMD instructions. However, the omission of the "Altivec" label was conspicuous by its absence. Furthermore, the presenter of the PowerPC 970 processor, Peter Sandon, had previously directed the design effort of a PowerPC compatible processor codenamed Gekko. Gekko had its own set of SIMD extensions, and this processor is now used by Nintendo in the Nintendo GameCube. As a result of these circumstances, it was unclear at the outset whether the 162 specialized SIMD instructions were indeed Altivec-compatible or not. During the presentation of the PowerPC 970 processor, Peter Sandon revealed that the Vector SIMD units were indeed Altivec-compatible. The vector SIMD ISA was co-developed by IBM and Motorola, but the "Altivec" name was trademarked by Motorola, so IBM could not use Motorola's trademarked name to describe a functionally compatible implementation of the same vector SIMD unit.

### Pipeline Depth

The PowerPC 970 processor has relatively long pipeline structures compared with the previous generation PowerPC G3 and G4 processors. There are 9 pipeline stages devoted to instruction fetch and decode, 5 to 13 pipeline stages are used for the out of order execution units. Simple integer instructions can execute in 5 stages, whereas more complex vector floating point instructions may take as many as 13 stages to complete execution. The somewhat surprising disclosure of the long 9 stage instruction and fetch decode pipeline was clarified with the work performed in this 9 stage fetch and decode process. As it turns out, some of the older PowerPC instructions are rather complex, and the solution that IBM had adopted in the POWER4 and the PowerPC 970 processor was an instruction cracking process whereby some complex instructions are cracked into simpler, more RISC-like instructions. These simpler instructions are then sent to the dispatch and execution units in the processor core. Although this process superficially resembles the micro-op or RISC-op decoding steps seen in various x86 processors, the PowerPC instruction set is not nearly as complex as the venerable x86 ISA, and only few instructions in the PowerPC ISA needs to be cracked into simpler instructions for execution.

### Preliminary Specifications

The processor will be manufactured using IBM's 0.13um SOI process with 8 layers of copper interconnects. The PowerPC 970 processor has already taped out, and parts exists in labs within IBM, and they are currently undergoing performance evaluation and system debugging work. IBM expects that the processor will be mature enough to be released to manufacturers for sampling in Q2 2003, and if all goes well, volume production should commence in the second half of 2003.

The processor should achieve a clock frequency of 1.4 GHz to 1.8 GHz on the 0.13um SOI process. Average power consumption at 1.8 GHz is expected to be 42 Watts with 1.3V Vcc. IBM estimates that the PowerPC 970 processor would be able to achieve a SPEC CPU 2000 INT score of 937 and a SPEC CPU 2000 FP score of 1051 at 1.8 GHz. These performance numbers compare well against Intel's current top end offering of a 2.8 GHz "Northwood" Pentium 4 processor. The 2.8 GHz Pentium 4 currently achieves a SPEC CPU 2000 INT score of 970 (base) and a SPEC CPU 2000 FP score of 976 (base). However, at the expected release date of 2H03, the PowerPC 970 would presumably have to compete against a 3.0+ GHz, Prescott based Pentium X processor. As a result, PowerPC 970 may not have the performance lead as measured with the SPEC CPU suite upon release in the second half of 2003. None the less, these preliminary SPEC CPU scores indicate that the performance of the PowerPC 970 processor will be far more than simply respectable, and the performance indicates a large leap above and beyond current PowerPC offerings from Motorola and IBM, respectively.

### Packet Based System Interconnect

One of the more interesting aspects of the PowerPC 970 processor is the system interconnect. Unlike the bi-directional processor busses seen on Intel IA-32 and IA-64 processor, or even the bi-directional point to point interconnects used on Alpha EV6 and AMD Athlon processors, the system interconnect of the PowerPC 970 processor are uni-directional, point to point, source synchronous interconnects that do not have to worry about bus loading factors or bus turnaround times, and the interconnect can wave-pipeline multiple number of bits of data on the wires concurrently. The most difficult part of such high frequency system interconnect may be the deskewing circuitry that would be required. In this case, the PowerPC 970 appears to have benefitted well from the POWER4 lineage, where the deskewing circuitry for a wave pipelined interconnect was previously disclosed by IBM.

The system interconnect on the PowerPC 970 has been designed to operate at an integer fraction of the CPU core frequency. At a CPU core frequency of 1.8 GHz, the system interconnect will operate at a frequency of 900 MHz. With two unidirectional 32 bit wide interconnects, one from the CPU to the companion system controller chip, the other from the companion system controller chip back to the CPU, the system interconnects can provide 3.6 GB/s of raw system bandwidth on each direction for an aggregate bandwidth of 7.2 GB/s. However, the unidirectional links must multiplex address and control information onto the same interconnects, and when these overheads are taken into considerations, IBM claims an effective peak data bandwidth of 6.4 GB per second.

### 32 Bit and 64 Bit Architecture

One question that has been asked is whether or not Apple would seek to adopt the PowerPC 970 processor and move to replace the PowerPC 7455 based processors currently shipping in Apple Macintosh computers. The consensus as presented by the analysts at Microdesign Resources is that there are but a few minor hurdles for Apple to accept the PowerPC 970 processor and use it as its main line desktop processor. One of the hurdles is the 32/64 bit question. Current PowerPC processors used in Apple Macintosh computers are 32 bit processors from Motorola, and the PowerPC 970 processor is a 64 bit processor. IBM addressed this point specifically by stating that the PowerPC 970 processor inherits the 32/64 bit mode switching mechanism from POWER4, and allows for relatively painless transition between 32 bit mode and 64 bit mode operation. Although 32 bit operating systems cannot operate on the PowerPC 970 processor without modification, the necessary modifications have been designed to be minimal. IBM also announced that the PowerPC 970 processor currently runs both 32 bit as well as 64 Linux in the testing labs within IBM. It is believed that with the minimal modification and support by the operating system, user mode 32 bit PowerPC applications can then run on the PowerPC 970 processor without modifications. With these assurances from IBM, it is believed that should Apple adopt the PowerPC 970, current 32 bit software would be able to run in a seamless fashion, while a 64 bit environment would also be available to the developers in the same system.

### System Chip Support

One of the more troublesome hurdles for Apple to overcome in the adoption of the PowerPC 970 processor may be the system engineering aspect of the processor. As described previously, the 4 byte wide unidirectional serial links may provide upwards of 6 to 7 GB of raw bandwidth per second. However, the specification of the ~ 900 MHz operations on the system board would require considerable investment into the system support chip. Moreover, the nature of the point to point interconnects means that to support a dual CPU system, the companion chip must be designed with the dual CPU SMP in mind, with dedicated channels devoted to each CPU. Furthermore, to support the high bandwidth available on the system interconnect, a dual channel PC2700 DDR SDRAM memory system would appear to be a minimum requirement to support a single CPU. Unless Apple can also obtain a low cost support chip from IBM, the PowerPC 970 processor would likely force the Apple Macintosh product lines to become even more upscale, and Apple would likely retain the use of the PowerPC G4 processors for the lower end iMac and eMac product lines.