The IBM PowerPC 970FX, a.k.a. G5 Processor

David Benham and Yu-Chung Chen

Department of Computer Science

10 October 2005

Abstract

This project is the research of the IBM PowerPC 970FX, also known as G5 processor. This report is for CS 466 Advanced Computer Architecture class taught in Fall 2005. In this report we present our research on the processor including the main specification. Several special features are also addressed. From section 1 to section 4 is written by David Benham and section 5 to section 7 is written by Yu-Chung Chen.

1. Introduction and History of PPC processor

The PowerPC processor line emerged as a product of the AIM alliance in the early 1990s. The AIM alliance was a cooperative effort between Apple, IBM, and Motorola. It's name is an acronym derived from the first letter of each company [1].

The AIM alliance was formed to develop a new RISC based architecture and computer platform along with a new operating system to compete with the widespread and immensely popular 'wintel' platforms consisting of Microsoft operating systems running on Intel based hardware [1]. Apple was looking to gain a new line of processors for their computers so they could replace their use of 68k CISC based chips, and IBM and Motorola sought a way to further their RISC processor design efforts [5].

Unfortunately, most of the original goals of the AIM alliance were never realized. It turned out that none of the companies could get an operating system developed for the new system and the new RISC based machine architecture and operating system efforts floundered and never gained widespread acceptance in the marketplace [1]. One product of the AIM alliance was the production of a RISC processor. The success of the AIM alliance in the production of the PowerPC processor was fueled, in part, by a Apple's decision to use the PowerPC processor in it's computers starting in 1994.

There is a great deal of speculation concerning the AIM alliance and why it never reached it's full potential or met all it's goals [2,3]. IBM and Motorola had differing philosophies for the direction of the PowerPC processor with respect to extending the instruction set to address DSP [6]. Unlike Apple, IBM and Motorola were interested in tailoring the processor to embedded application markets [3] and the embedded market was often more concerned about price and stability instead of cutting edge performance when it came to the PowerPC processor. Apple was also impatient with Motorola's ability to keep up with ever increasing performance demands from the marketplace [4].

Apple's recent decision to begin using Intel based processors in their computers has effectively dissolved the partnership.

2. PowerPC 970 overview

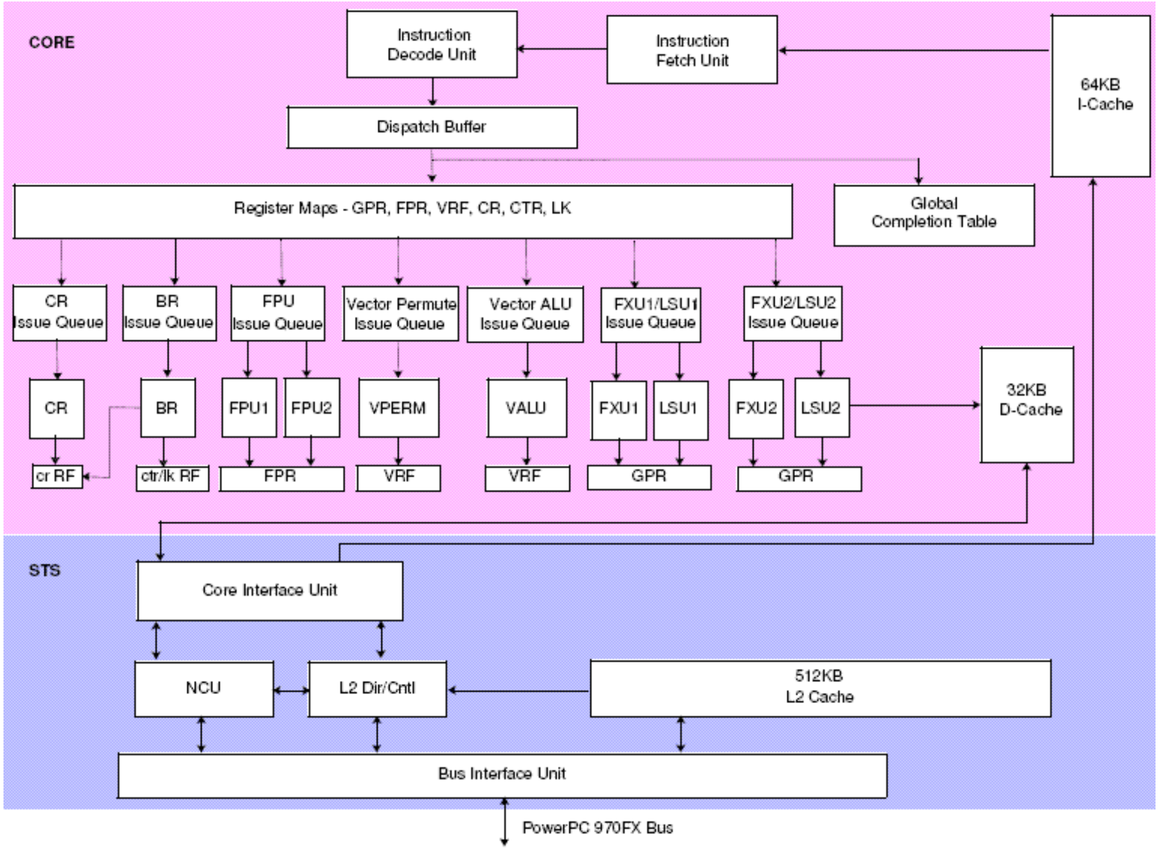
The PowerPC 970 is a 64 bit RISC processor produced by IBM. The PowerPC 970 is commonly referred to as the PowerPC G5 , a moniker coined by Apple. The chip is a 64 bit processor, but is capable of supporting 32 bit PowerPC code natively.

Early PowerPC 970 chips were constructed with a 130 um and silicon-on-insulator (SOI) manufacturing process with a die size of 121 mm2 [1]. The 970FX, essentially version 2 of this chip, was updated and manufactured using a SOI and strained silicon 90um process with a die size of 65mm2 [2]. Both versions of chip contain approximately 58 million transistors [2]. The processor contains 32 64 bit general purpose registers, 32 64 bit floating point registers, 32 128 vector registers for use with the Altivec instructions (a SIMD multimedia instruction set), and a number of special use registers [3]. Clock speeds for the 970FX currently run at speeds up to 2.5 Ghz [2]. IBM is currently working on the 970MP, which is a dual core processor that can obtain speeds up to 3.5 Ghz [4].

There are many significant improvements in the 970 over the previous version of the PPC processor, the PowerPC 7400 (G4) line of processors. The obviuos is an increase in addressible memory and operand size between the two chips when increasing from 32 bits to 64 bits. The 970 has a virtual address rance of 64 bits, although in practice, the physical address space is only 42 bits wide, allowing access of up to 4 terabytes of memory. [5]

The 970 also introduced an enhanced SIMD multimedia instruction set, a greatly increased pipeline depth, and a significant increase in bus speed over the G4.

Chip overview



Power PC 970 architecture (IBM Image [1])

From a high level, the PowerPC 970 consists of three components, the Core, Storage Subsystem, and pervasive functions.[1] The core contains all the functional units, registers, and the L1 caches. The Storage subsystem contains the L2 cache as well as all the logic required for it's maintenance and interfaces to both the core and the processor bus.

The processor contains 512 KB of L2 cache and 96KB of L1 cache. The L1 cache is further broken down into two segments, a 32 KB segment (D-cache) for the data and 64 KB segment (I-Cache) for instructions.

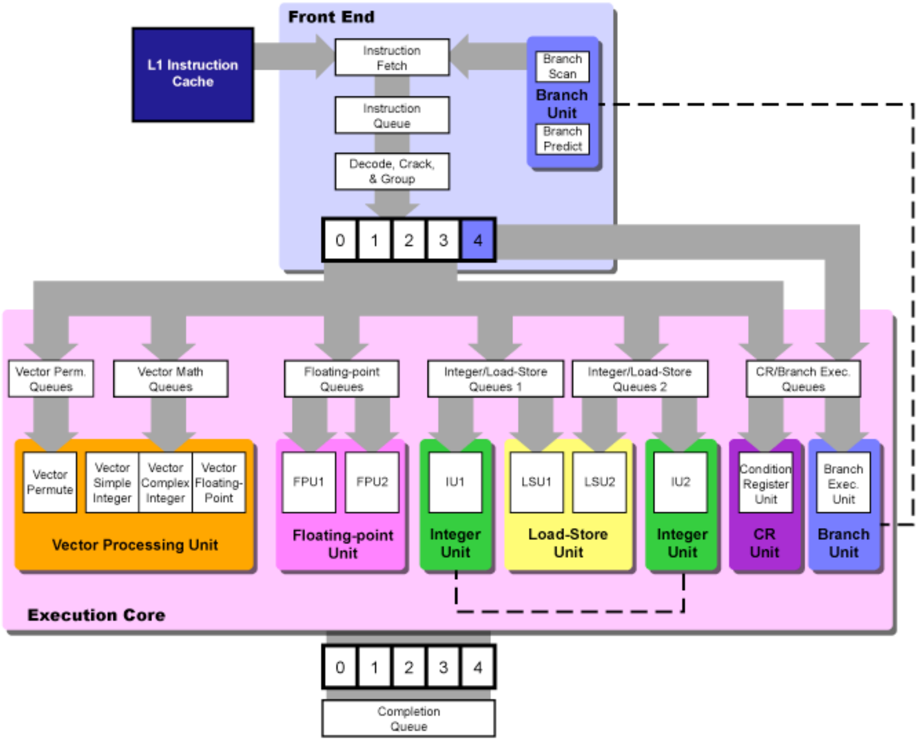
The I-Cache is direct mapped and contains 128 byte lines, for a total of 512 lines. Each line is broken down into 32 byte sectors. The L1 cache interface to the L2 cache is

32 bytes wide, and when cache misses are encountered, the L1 cache is loaded from L2 by using a "critical sector first reload policy" in order to fetch the required sector as quickly as possible.[1] The I-Cache supports one read or one write per cycle. [1]

The D-Cache is two way set associative and it uses a LRU replacement algorithm for selections after cache misses. When data is not present in the cache for a write, a store through policy is used, so there is no allocation of a entry for the missing line. [1] The D-Cache supports two reads and one write per cycle. [1]

Just above the L1 cache sits a prefetch queue. It contains 4x128 byte entries. The prefetch unit can be controlled by either the hardware itself or by software instructions.[1] The prefetch queue analyzes program flow by working closely with the LSU (load store units) and instructs both the L1 and L2 caches on data to load that is anticipated for future use. [1]

After instructions have been fetched, they move to the decode unit. Here, most instructions are broken down, or cracked, into smaller internal operations (IOPs) for easier processing and to aid in increasing some instruction level parallelism[2]. Most PPC instructions translate into a single IOP, however a handful of instructions translate into 2 or more IOPs [3].



Power PC 970 functional units(Image courtesy of Jon Stokes[2])

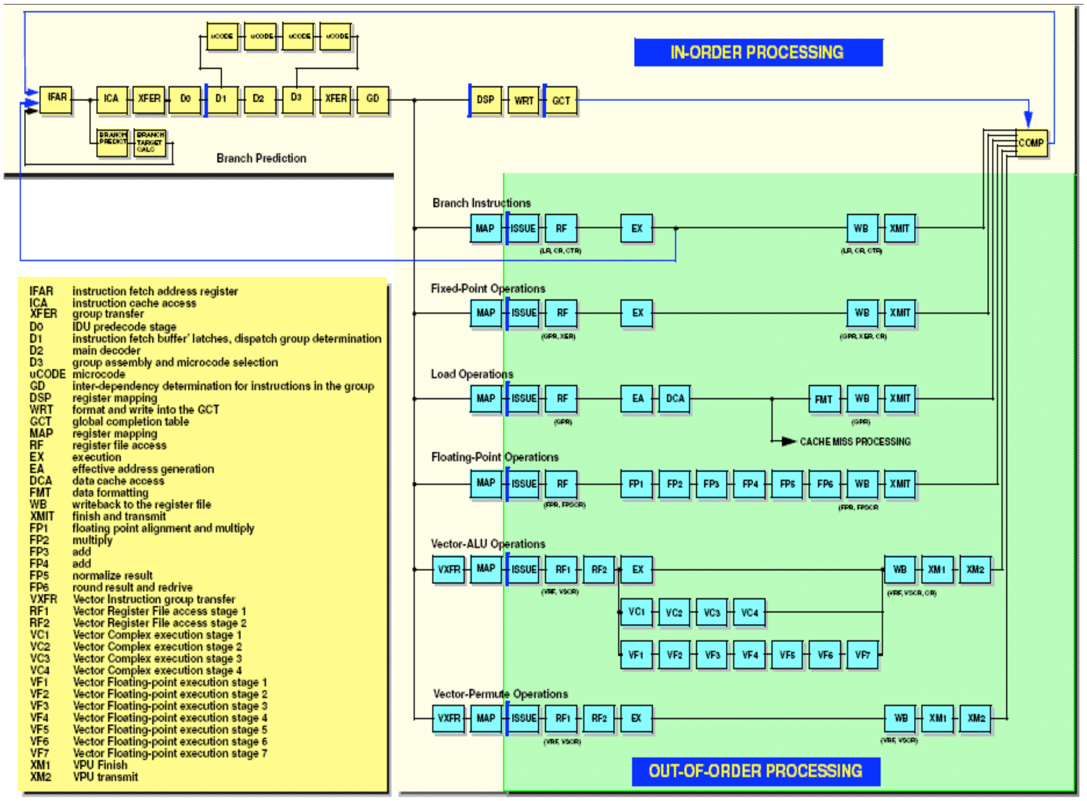
Instructions are also categorized into dispatch groups for relay to the execution pipelines. Dispatch groups consist of 5 IOPs, 4 of which can be an instruction of any type, with the last slot reserved exclusively for branch instructions. Instructions are tracked by dispatch group as they proceed through the pipeline and the state of the processor is tracked at the boundaries of each group. Certain instructions are not allowed to be in a dispatch group with other instructions and must be executed alone.

Once instructions have been placed in a dispatch group, they are sent to one of 6 issue queues before being sent to the execution pipelines. After execution is complete, the Group Completion Table (GCT), which is the equivalent of a reorder buffer, aids the processor in reassembling the operations. The GCT tracks groups of instructions, instead of individual instructions. Tracking groups instead of instructions reduces some of the overhead, although there is added complexity because of the rules necessary when forming the instruction groups.

3. Pipeline

The following are the number of stages for the different instruction types supported in the 970FX.

- 16 stages for fixed-point register-register operations
- 18 stages for load and store operations
- 19 stages for fixed-point VALU operations
- 19 stages for vector permute operations
- 21 stages for most floating point operations
- 22 stages complex-fixed VALU operations
- 25 stages for floating point operations VALU operations



IFAR	instruction fetch address register
ICA	instruction cache access
XFER	group transfer
D0	IDU predecode stage
D1	instruction fetch buffer' latches, dispatch group determination
D2	main decoder
D3	group assembly and microcode selection
uCODE	microcode
GD	inter-dependency determination for instructions in the group
DSP	register mapping
WRT	format and write into the GCT
GCT	global completion table
MAP	register mapping
RF	register file access
EX	execution
EA	effective address generation
DCA	data cache access
FMT	data formatting
WB	writeback to the register file
XMIT	finish and transmit
FP1	floating point alignment and multiply
FP2	multiply
FP3	add
FP4	add
FP5	normalize result
FP6	round result and redrive
VXFR	Vector Instruction group transfer
RF1	Vector Register File access stage 1
RF2	Vector Register File access stage 2
VC1	Vector Complex execution stage 1
VC2	Vector Complex execution stage 2
VC3	Vector Complex execution stage 3
VC4	Vector Complex execution stage 4
VF1	Vector Floating-point execution stage 1
VF2	Vector Floating-point execution stage 2
VF3	Vector Floating-point execution stage 3
VF4	Vector Floating-point execution stage 4
VF5	Vector Floating-point execution stage 5
VF6	Vector Floating-point execution stage 6
VF7	Vector Floating-point execution stage 7
XM1	VPU Finish
XM2	VPU transmit

Pipeline(IBM Image [1])

The pipeline of the 970FX consists of a master pipeline and multiple execution pipelines. The master pipeline processes instructions in order, and is responsible for the fetching, decoding, and dispatching instructions execution pipelines.

The execution pipelines support out of order instruction execution for up to 10 operations:

- Two load or store operations
- Two fixed-point register-register operations
- Two floating-point operations
- One branch operation
- One condition register operation
- One vector permute operation
- One vector ALU operation

The 970FX dispatcher can issue 5 instructions per clock cycle to the execution units and instructions are issued speculativly based on earlier branch predictions. In the event of a misprediction or exception, the instructions and thier results are discarded. [1]

4. Branch Prediction.

The 970FX is capable of predicting two branches per clock cycle. Of the 8 instructions the processor is able to fetch from memory per clock cycle, it can predict branch targets for two of those instructions. Branch prediction is accomplished in the 970 using several different methods. Three history tables are used to aid in branch prediction:

- Local predictor table
- Global predictor table
- Selector table

The *local predictor* table is a 16k table, indexed by branch address. Each bit in the table simply indicates if the branch should be taken or not taken.

The *global predictor* table stores context information about instruction groups. As instructions are loaded from memory, information about the group of instructions loaded is recorded in the global predictor table. The information stored is an 11-bit vector, used to trace the path of execution of instruction groups [1]. Information is retrieved and stored in the table by hashing this 11-bit vector with the instruction address. Using this information, branch prediction is able to trace the execution path of previously loaded instructions and see if subsequent instruction loads were done from a sequential cache sector. For a given instruction group, if the next group was not loaded from a subseqential cache location, then the branch prediction algorithm will predict taken for this branch. The predictor is a single bit and the size of the global predictor table is 16k.

The *selector table* is a 16k table that tells the branch prediction scheme if it should use either the Local or Global predictor table, based on which has performed better for a particular branch.

In addition to these tables, software has the option of setting branch instruction in the branch instruction to override default branch instruction behavior. There are three cases where the branch prediction of the processor should be overridden. [1]

1. When the branch probability is extremely high. If a branch is taken only in rare occasions, such an exception, tell the processor how to handle the case.

2. Branches that end a segment of code when obtaining a lock. In these cases, the code wants to tell the compiler to predict the branch as not taken, because it is likely a lock will be obtained.
3. Intentionally force a misdirection to aid in instruction pre fetching from memory. A branch misprediction is a costly event, but there are certain cases this overhead is less than a cache miss.

It is also possible for software to completely disable all the branch prediction capability of the 970FX if necessary.

5. VMX/AltiVec

VMX is PowerPC 970FX's vector processing unit. It is also called "AltiVec" by Motorola (or Freescale, a spin-off from Motorola) and "Velocity Engine" by Apple. In the early days, the floating point processing unit (FPU) is built as a co-processor for x86 CPUs. With technology advances, now we can put floating-point process unit (FPU) in the same die of CPU. Now IBM puts VPU hardware in PPC 970FX.

5.1 What is vector processing?

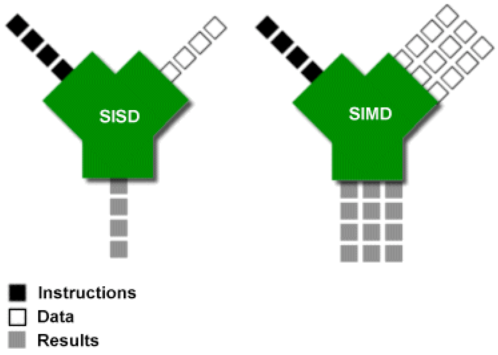
Most of the CPUs we used today are scalar ones. That means the CPU can only process the data one at a time. Vector CPUs are more seen in scientific computing environments, where large amount of number crunching computation is required. We can see a lot of vector supercomputers in 1980s to 1990s. Vector means an array of multiple data in the same data type. Vector CPUs can do computation on a vector in one single instruction. Depends on the length of the vector a processor can work on, there are long vector processors and short vector processors. If the processor can work on a vector containing up to hundreds of data elements, we will call it 'long-vector' processor. One of the earliest vector processor is ILLIAC IV, which is developed in University of Illinois in 1970s [1,7]. There are several vector-based supercomputers, including famous Cray series.

PowerPC 970FX's VMX extension is a 'short-vector' processing unit, which can do vectors that contains 4, 8, or 16 elements [4]. Compared with 'long-vector' processors, putting extension instructions and hardware with a general-purpose scalar central processing unit will have more flexibility. Because it will fit most of general users' need, and that is why there are almost no 'pure-vector' general purpose central processing units in commodity market nowadays.

5.2 Single Instruction Multiple Data vs. Instruction Level Parallelism

There are several different ways to exploit the parallelism in instruction set architecture and central processing unit hardware design. One way is through instruction level parallelism (ILP). The main idea in ILP is trying to execute as many independent instructions as possible at the same time (if required hardware is available). Pushing further, architects will try to put in more available hardware forming a superscalar processor. This is doing parallel with instructions [7].

An alternative parallel mechanism is doing parallel with data. With 'Vector Processing Unit' and extended vector instruction set, PPC 970FX can operate a vector of data (of the same type) with a single instruction. The difference between 'Single Instruction Single Data' and 'Single Instruction Multiple Data' can be visualized as:

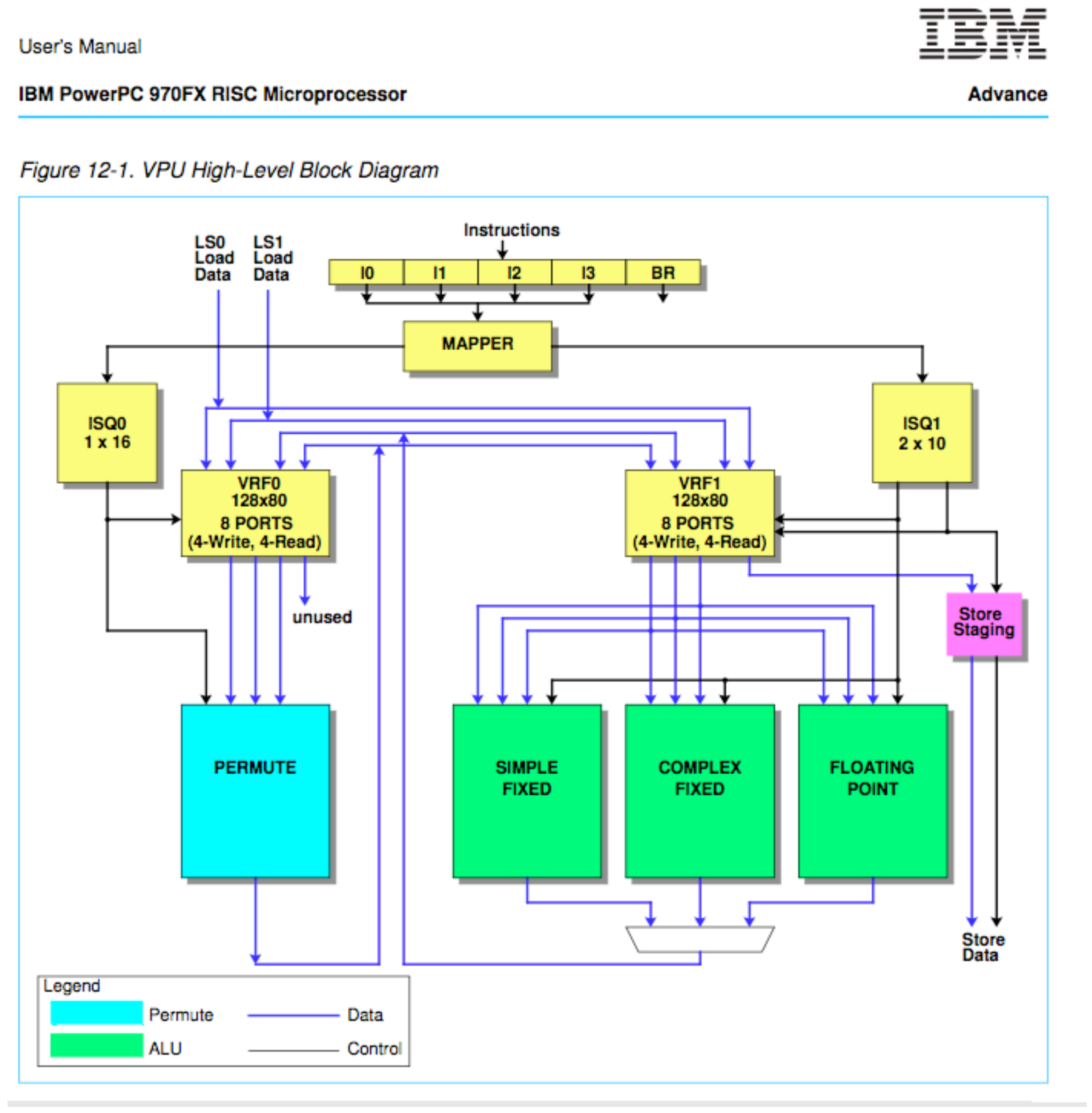


From: http://arstechnica.com/articles/paedia/cpu/simd.ars/1

Of course, this alternative to exploit loop-level parallelism will have some side effects on the whole system design. For example the vector processing unit will be pipelined and have a relatively longer latencies, but it can be overlapped [4]. And the 'data-hungry' nature will also make architects design a faster and higher bandwidth between the processor and memory system (L1/L2 cache and RAM). Although in today, ILP based processor almost completely replaced vector-based ones, but we are seeing more and more vector extensions been used in areas like graphics, digital signal processing, and multimedia applications [7].

5.3 PowerPC 970FX Vector Unit Specification

PowerPC 970FX's Vector processing unit is a SIMD extension to PowerPC instruction set. In hardware, the high-level block diagram looks like this:



User's Manual

IBM PowerPC 970FX RISC Microprocessor

Advance

Figure 12-1. VPU High-Level Block Diagram

PPC 970 VPU Block Diagram

From: IBM PowerPC 970FX RISC Microprocessor User's Manual [4]

Most of the detailed specification can be found in the reference user's manual [4]. Here we will look at several key features and components of the VPU.

All data path width is raised to wider design of 128 bits, which means in each vector (register file), it can either store 4 single-precision (32-bit) floating point words, 4 32-bt integer words, 8 16-bit integer half-words, or 16 8-bit integer bytes. There will be two separate sets of ‘Vector Register Files’, one for ‘Vector Permute Unit’ and other for ‘Vector Arithmetic Logic Unit’. Each set consists of 80 renamed registers with 4 write ports and 4 read ports. The execution paths of these two functional units (Vector Permute Unit and Vector ALU) are two fully pipelined and independent. Both of these two units will have their own issue queue, so VPU can do dual VMX instruction issues: one for ‘Vector ALU’, and one for ‘Vector Permute Unit’ [4]. One thing worth mention, in vector computation, each vector needs to be ‘prepared’ (pack) before entering the vector functional unit. The ‘Vector Permute Unit’ will serve with such requirements.

Digging deeper, the ‘Vector Arithmetic Logical Unit’ can be divided to three kinds of operations (paths): Floating point (7-stage execution), Simple fixed (1-stage execution) and Complex fixed (4-stage execution). Depends on the input vector’s data type, they can be further partitioned into: 4 separate ALUs for 32-bit integers, 4 separate ALUs for single-precision floating points, 8 separate ALUs for 16-bit integers, or 16 separate ALUs for 8-bit integers. The largest added in the vector ALU is 32 bits wide, and the largest multiplier array is 24 bits wide, for single precision floating point [7].

The ‘Vector Permute Unit’ will take 1-stage pipeline execution (within the whole execution pipeline, the whole pipeline will have 19 stages). ‘Vector Permute Unit’ is a highly flexible byte manipulation engine. It can prepare the vector operands with pack and unpack operations from and to scalars data elements. Furthermore, if you need to combine data elements in multiple vectors to form a new vector, this particular execution unit will allow you to do that without going all the way to memory load and store with multiple instructions [4]. This will save a lot of clock cycles going back and forth.

Vector processing relies heavily on memory/cache access and bus performance, because as a vector processing unit, it tends to fetch more data in each instruction. If it needs to get the data all the way through RAM in each instruction, the performance will drop drastically. This is why trying to fit blocks of vector data into the L1/L2 cache memory to get better performance is one general rule of thumb in vector programming. Increasing the bus speed and the bandwidth between CPU and memory are also important. In current computers based on PPC 970FX, for example Apple PowerMac Dual G5 machine, each CPU will have an independent 1.35GHz, 64-bit bidirectional Double Data Rate (DDR) frontside bus maximizes throughput between CPU and the rest of the system. And this will achieve up to 10.8GB per second of aggregate bandwidth (bi-directional) for each processor [6].

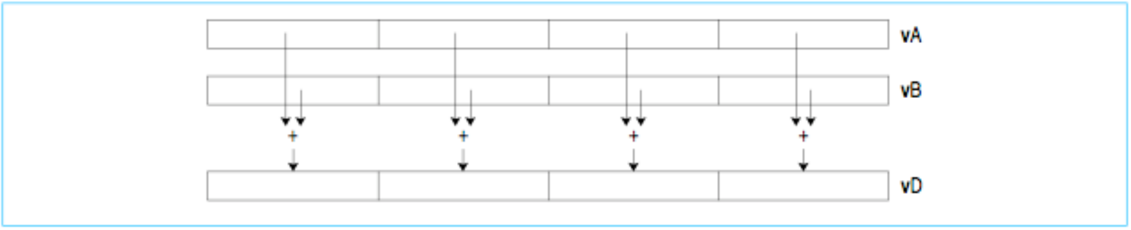
5.4 Vector instruction set

VMX (AltiVec) adds 162 new vector instructions into PowerPC instruction set architecture. Each instruction length is mapped to fixed-sized with 32 bits and word-aligned. A vector instruction can use up to 3 source-vector registers, and 1 destination vector register. The execution path in VPU is also fully pipelined. For permute instructions, it will take 1 stage execution, totally 19-stages pipeline throughout the whole execution pipeline. For Vector ALU, depends on what kind of operations (simple fixed, complex fixed, or floating-point), it will be 1, 4, and 7 stages execution. And that will be 19, 22, and 25 stages pipeline [4].

Vector ISA will allow ‘Intra-element’ operation and ‘Inter-element’ operation [4]. Each will also include arithmetic and non-arithmetic operation. This leads up to 4 kinds of combinations. Two examples are shown below.

In ‘intra-element’ operation example, data elements of the same index in two source vectors will be used as computing operands and the result will be stored at the same index of the destination vector. An example of ‘intra-element’ operation is Vector Add Signed Word Saturate (vaddsws) instruction. In ‘vaddsws’, 4 signed integer elements in vector register ‘vA’ are added to the corresponding 4 elements in vector register ‘vB’ and the four sums elements are stored in vector register ‘vD’ in corresponding indices.

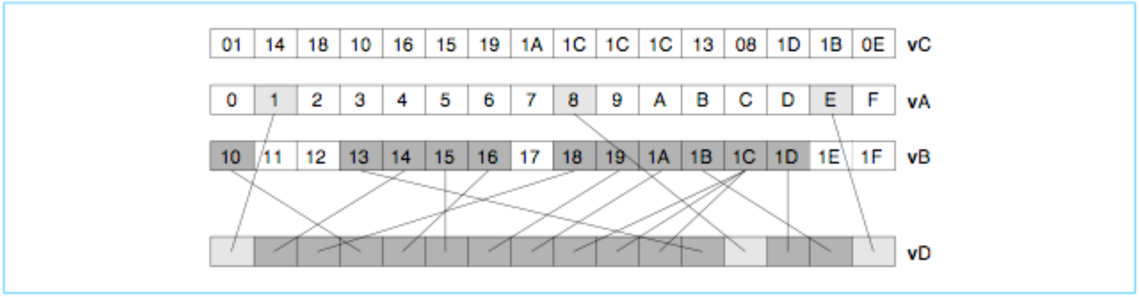
Figure 1-4. Intraelement Example, **vaddsws**



From p.30 of IBM’s PowerPC Microprocessor Family: Vector/SIMD Multimedia Extension Technology Programming Environments Manual

In ‘inter-element’ example, elements of different indices in difference source operands are fetched independently and used, then the result element is also stored in different index. And example instruction is ‘Vector Permute’ (vperm). ‘vperm’ will grab any elements from source ‘vA’ or ‘vC’, and copy to destination vector register ‘vD’. ‘vC’ is used as the target indices of the elements to be fetched.

Figure 1-5. Interelement Example, **vperm**


01 14 18 10 16 15 19 1A 1C 1C 1C 13 08 1D 1B 0E
vC
0 1 2 3 4 5 6 7 8 9 A B C D E F
vA
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
vB
vD

From p.30 of IBM’s PowerPC Microprocessor Family: Vector/SIMD Multimedia Extension Technology Programming Environments Manual

5.5 What is VMX good for?

With the vector processing capabilities, what is the main application of such powerful hardware instruction set architecture? To answer this question, we must first look what areas will VMX contribute most. The characteristic of capable operating multiple scalar data element in the same time will accelerate basic low-level operations like memory copies, string comparison, and page clears. For those applications that need complex mathematics (matrices related), science (those involve with high degree of linear systems), and graphics manipulation on great amount of stream-lined homogeneous data, VMX can also leverage the flexible vector hardware extension to provide the whole CPU better performance. An example of using VMX to increase the mathematical matrix operation is shown in [3]. Although the machine used in the example is only an Apple 400MHz G4 with AltiVec extension, but the most complex matrix multiplication case is shown boosting from 84 MFLOPS to 384 MFLOPS (Mega-FLoating-point Operation Per Second).

An alternative approach toward these application areas is deploying a general CPU with multiple special-purpose digital signal processors (DSP) [5]. I agree with [5] states that for high-level hardware developers, this also implies multiple and complicated hardware architectures and chaining toolsets (for embedding systems). On the contrary, the PowerPC ISA plus vector extension provides a more flexible and homogeneous developing environments.

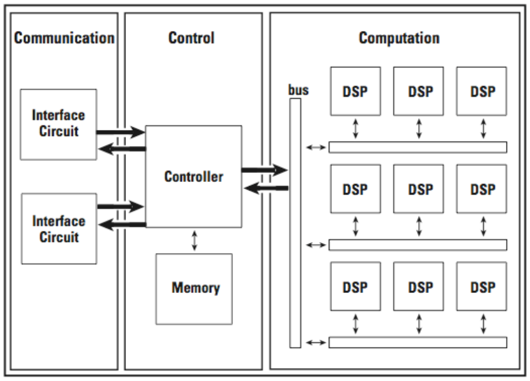


Figure 5. Typical controller plus DSP system

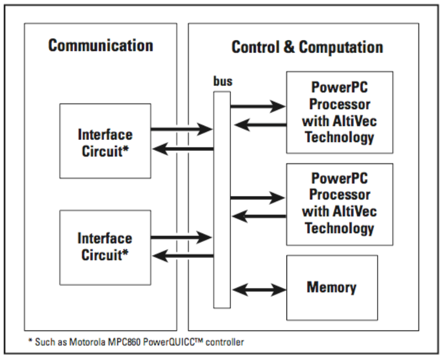
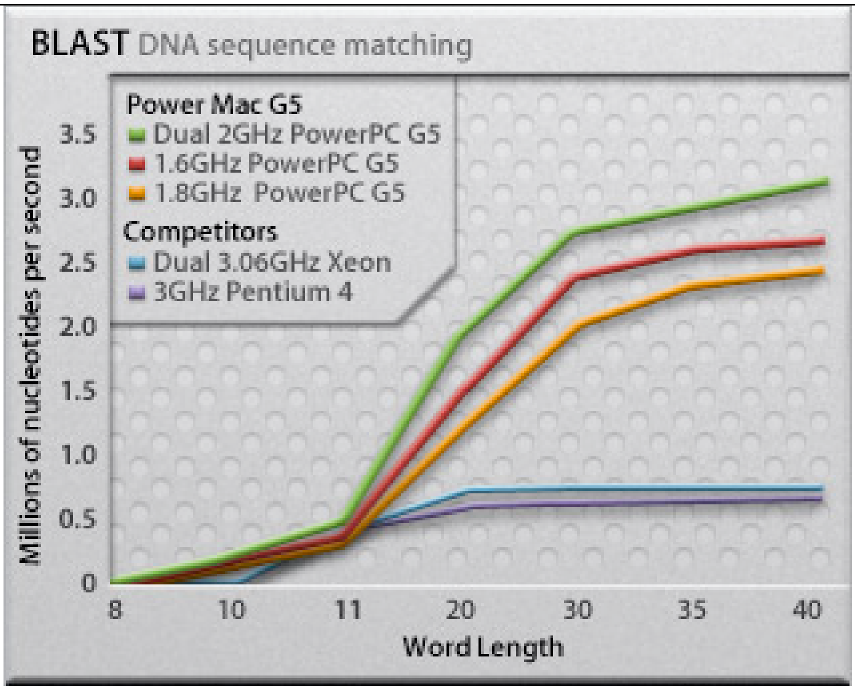

Communication
Control & Computation
bus
Interface Circuit*
Interface Circuit*
PowerPC Processor with AltiVec Technology
PowerPC Processor with AltiVec Technology
Memory
* Such as Motorola MPC860 PowerQUICC™ controller

Figure 6. System using multiple PowerPC processors with AltiVec technology, sharing a common bus bridged to shared memory

Comparisons between RISC + multiple DSP vs. RISC + AltiVec VPU

From Motorola (Freescale) AltiVec Technology whitepaper [5]

In hardware, VMX technology is used in multi-channel modems, image and video-processing systems, scientific array processing systems, and IP telephony appliances [5]. For the software side, Apple/Genentech AltiVec-optimized BLAST implementation running in PPC970FX G5 CPU outperformed the NCBI (National Center for Biotechnology Information) version up to 5 times in the best case [6].



Data from: http://www.apple.com/lae/powermac/performance/ [6]

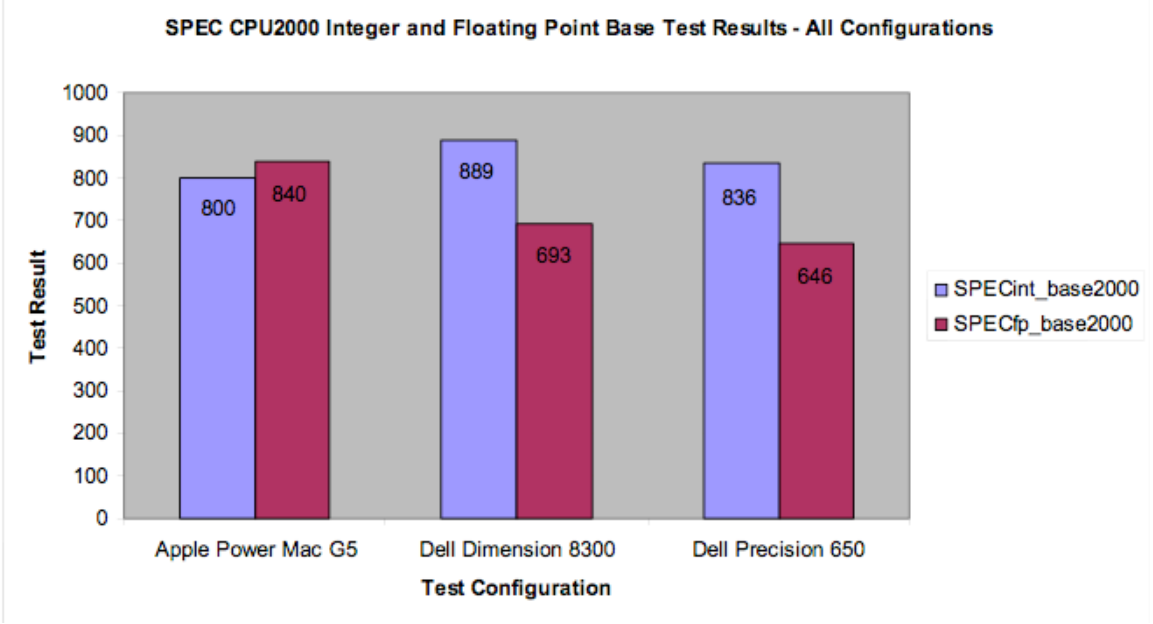
6. Benchmarks

So, how fast is PowerPC 970FX processor? In this section we will look at several benchmarks ranging from standard benchmark suite, a day-to-day desktop program, and several scientific computation-intensive programs.

6.1 Standard SPEC benchmark suite

From the data I collected, there are actually two copies of SPEC results; one is from the Apple/Veritest released result [1] using Apple's PowerMac Dual 2GHz G5 machine running MacOSX operation system. And the other is released from SPEC [2][3] using IBM eServer BladeCenter JS20 with 2.2GHz PowerPC 970 processors, running IBM AIX 5L V5.3.

In Apple/Veritest's benchmark report, 2.0GHz G5 machine's scores is not as good as Intel 3.06GHz Xeon machine (800 vs. 836). But in the SPECfp floating-point benchmark, 2GHz G5 outperformed 3.06GHz Xeon machine with 840 to 646. One possible speculation toward the result might be the dual floating-point processing units in each PowerPC 970 CPU and the higher bus speed and bandwidth between CPU and the rest of the system.



SPECint and SPECfp scaled score release by Apple/Veritest [1]

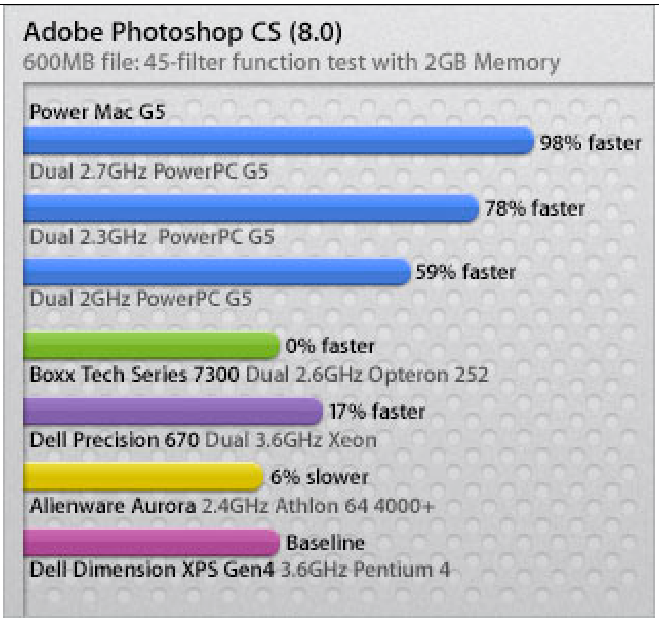
However, in the report released in SPEC.org, we cannot find Apple's PowerMac machines. Instead, an IBM eServer BladeCenter JS20, which is based on PowerPC 970 CPU, running IBM AIX operation system is listed. And the numbers shown (see below table) are not as impressive as those shown in Apple/Veritest report. The PPC970 CPU machine is still looked faster, but the difference is not so dramatic.

IBM eServer BladeCenter JS20 (2200 MHz, 1 CPU), 1 core, 1 chip, 1 core/chip	
SPEC CPU2000 Base: 986 Peak: 1040	SPEC CFP2000 Base: 1178 Peak: 1241
Dell PowerEdge 1750 Xeon 3.06GHz	
SPECint2000 Base: 1031 Peak: 1067	SPECfp2000 Base: 1030 Peak: 1044

Some suggested [11] that Apple tweaked the comparing machines and configurations to make their machine's benchmark looked better. Something like, running the benchmark with optimized code in G5 machine but others. Apple denied that however. But this is just standard set of benchmarks. So next we will look at several possible real-world application benchmarks.

6.2 Real-life applications

In the first case of Photoshop timing scenario, with 45 image processing filters upon 600MB image file in 2GB memory in all testing machines, Dual 2.7GHz G5's 98% faster than Dual 2.6GHz AMD Opteron.

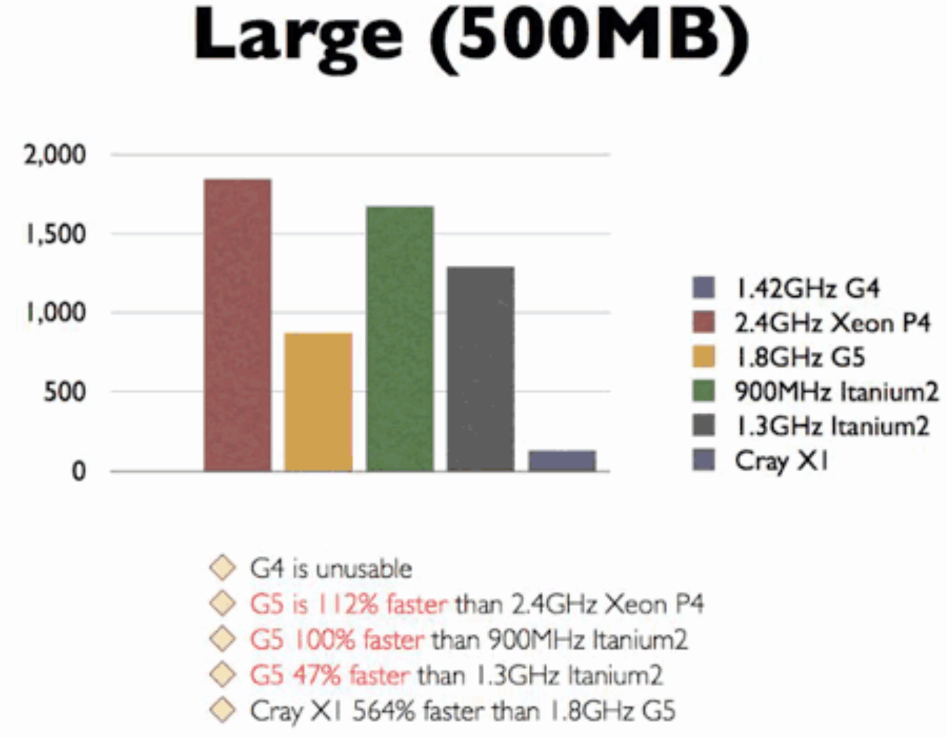


From http://www.apple.com/powermac/performance/ [5]

It seems too good to be true. One obvious reason could be that, at the time of the tests were conducted, there is no valid working version of 64-bit operation system and 64-bit version of Photoshop available in Opteron platform. So there is no optimized version of Photoshop available for Opteron hardware platform. On the contrary, at the very first moment Apple PowerMac G5 released, Adobe provided an G5 optimized version of the software. Some image filters were even optimized to G5's VMX (AltiVec) vector extension, which will accelerate the processing speed noticeably.

In scientific computation domain, I will look at two computational-hungry beasts, computational fluid dynamics, and structured biology.

According to the benchmark results [6] with computational fluid dynamics laboratory code by Dr. Sean Garrick of University of Minnesota, for cases that need a lot of memory. The G5 machine even out-performs Itanium2. Two observations on this user's own benchmark program had been made. One is that the code is very floating-point computation intensive, and the other is, G5 relatively performs better in cases which more memory space is required [6]. The independent high clock rate bus does help keeping the CPU run at its full speed [12]. One other worth mentioned is that Itanium2 machines are generally much expensive than the G5 machines.



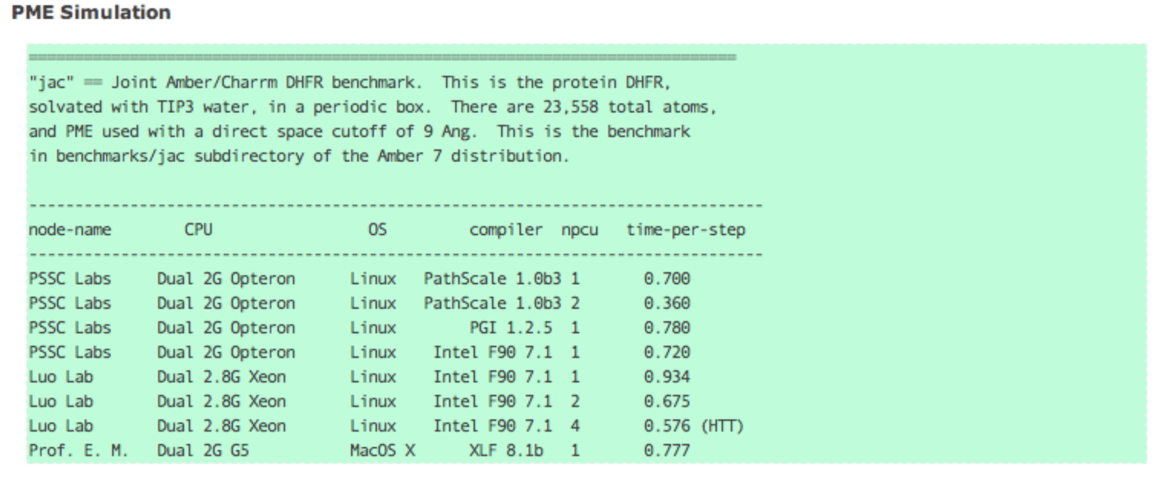
- G4 is unusable
- G5 is 112% faster than 2.4GHz Xeon P4
- G5 100% faster than 900MHz Itanium2
- G5 47% faster than 1.3GHz Itanium2
- Cray X1 564% faster than 1.8GHz G5

Data from:

http://www.xlr8yourmac.com/G5/G5_fluid_dynamics_bench/G5_fluid_dynamics_bench.html

[6]

At last, my friend Mengjuei Hsieh who majors in structure biology in University of California at Irvine did a serial test runs on their laboratory's major simulation software for molecular simulations.



PME Simulation

```
==============================================================================
"jac" == Joint Amber/Charrm DHFR benchmark.  This is the protein DHFR,
solvated with TIP3 water, in a periodic box.  There are 23,558 total atoms,
and PME used with a direct space cutoff of 9 Ang.  This is the benchmark
in benchmarks/jac subdirectory of the Amber 7 distribution.

--------------------------------------------------------------------------------
node-name        CPU                  OS         compiler  npcu   time-per-step
--------------------------------------------------------------------------------
PSSC Labs     Dual 2G Opteron     Linux    PathScale 1.0b3 1       0.700
PSSC Labs     Dual 2G Opteron     Linux    PathScale 1.0b3 2       0.360
PSSC Labs     Dual 2G Opteron     Linux         PGI 1.2.5  1       0.780
PSSC Labs     Dual 2G Opteron     Linux     Intel F90 7.1  1       0.720
Luo Lab       Dual 2.8G Xeon      Linux     Intel F90 7.1  1       0.934
Luo Lab       Dual 2.8G Xeon      Linux     Intel F90 7.1  2       0.675
Luo Lab       Dual 2.8G Xeon      Linux     Intel F90 7.1  4       0.576 (HTT)
Prof. E. M.   Dual 2G G5          MacOS X      XLF 8.1b    1       0.777
```

From: http://apple.sysbio.info/~mjhsieh/archives/000295.html [8]

From his results, we can see that G5’s performance is better than 2.8GHz Xeon machines, but not as good as 2GHz Opteron machines with ‘proper’ Fortran compiler. But the margin is not a great gap.

In summary, I think it can be concluded into two words, ‘it depends’. In my former workplace, every time before a major computer hardware purchase, the hardware’s evaluation must be went through a series of production code test runs. The best way to evaluate a new platform is to run your own daily programs on it and compare the result with that in your prior hardware platform. Also, the temporal performance will not be the only one metric of evaluation. One of our recent examples is that a Hewlett-Packard Superdome was tested and the performance benchmark also looked great, but the computation result is not inconsistent with the theoretical outcomes! The bottom-line, run your daily-use software programs, and see if the result is correct and how well it runs.

7. “Power Everywhere”

“POWER” stands for “Performance Optimization With Enhanced RISC”. The “Power Architecture”(PA) is not only being used on desktop computer systems. As discussed in the first section, PA is used in wide range of computer and electronic devices. With the history trace from the first IBM 801 CPU to the latest Power5 and Power6 in the future, we can see the same or slightly modified architecture is used in devices from supercomputers that take up to several basketball courts to power-saving embedded systems that is packed and sent to Mars [1, 4]!

Hardware	Description
Supercomputers 6 out of top 10 supercomputers [5] in top500.org are from IBM or using POWER architecture	Current (June 2005)) No.5 is actually using 4,812 PowerPC 970 CPUs to achieve R_{max} of 27910 FLOPS
IBM eServer BladeCenter JS20 [6]	IBM PowerPC 970FX
Apple PowerMac G5 [7]	IBM PowerPC 970FX
PlayStation3 Xbox 360	IBM Cell processor
Nintendo GameCube [1]	IBM PowerPC 750CXe processor
Mars rover, Spirits and Opportunity [4]	Radiation hardened 32-bit RISC CPU RTOS VxWorks

From IBM’s processor roadmap, it has the tendency of covering from performance-per-mW to performance-per-GHz. And with the recently released “Power Everywhere” policy in open license, almost all of the technology details and specifications are open to the public. It is a reasonable assumption that we can see more and more devices or computers using POWER technology and architecture and architecture been developed and merchandized in the near future.

Bibliographies

Introduction and History of PPC processor

1. AIM_alliance - AIM Alliance, http://en.wikipedia.org/wiki/
2. Mac Musings, Honestly, Dan Knight, http://lowendmac.com/musings/honest.html
3. Apple's Power Failure, Paul DeMone, http://www.realworldtech.com/page.cfm?ArticleID=RWT051400000000
4. Motorola slammed with PPC G4 supply limitation allegations, Tony Smith, http://www.theregister.co.uk/2000/03/07/motorola_slammed_with_ppc_g4/
5. PowerPC, http://en.wikipedia.org/wiki/PowerPC
6. Motorola And IBM Split On Direction For PowerPC, Anthony Cataldo and Loring Wirbel, http://www.techweb.com/wire/story/TWB19980511S0002

PowerPC 970 overview

1. Inside the IBM PowerPC 970, Jon Stokes, http://arstechnica.com/articles/paedia/cpu/ppc970.ars/1
2. Understanding 64-bit PowerPC architecture, Cathleen Shamieh, http://www-128.ibm.com/developerworks/library/pa-microdesign/
3. PowerPC Registers and Addressing Modes. Apple computer developers document. http://developer.apple.com/documentation/DeveloperTools/Reference/Assembler/PPCInstructions/chapter_6_section_2.html
4. PowerPC 970, http://en.wikipedia.org/wiki/PowerPC_970

   Power Mac G5, http://en.wikipedia.org/wiki/Power_Mac_G5

Chip overview

1. IBM PowerPC 970FX RISC Microprocessor User’s Manual, V1.5
2. Inside the IBM PowerPC. Jon Stokes. 970http://arstechnica.com/articles/paedia/cpu/ppc970.ars/4

3. G5 Revealed, Introduction to the PowerPC 970. Daniel A. Steffenhttp://www.maths.mq.edu.au/~steffen/talks/ppc970.pdf

Pipeline

1. IBM PowerPC 970FX RISC Microprocessor User’s Manual, V1.5

Branch Prediction.

1. IBM PowerPC 970FX RISC Microprocessor User’s Manual, V1.5

Basics

1. IBM PowerPC 970FX RISC Microprocessor User’s Manual, V1.5 http://www-306.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_970_and_970FX_Microprocessors
2. Understanding 64-bit PowerPC architecture http://www-128.ibm.com/developerworks/library/pa-microdesign/?ca=dgr-lnxw07UnderstandPower
3. PowerPC Architecture Book http://www-128.ibm.com/developerworks/eserver/articles/archguide.html?S_TACT=105AGX16&S_CMP=DWPA

VMX/AltiVec

1. Vector processor http://en.wikipedia.org/wiki/Vector_processor
2. Improving Fluid Dynamics http://www.apple.com/hotnews/articles/2002/10/craighunter/
3. The AltiVec Difference by Craig Hunter http://www.macdevcenter.com/lpt/a/1692
4. IBM PowerPC 970FX RISC Microprocessor User’s Manual, V1.5 http://www-306.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_970_and_970FX_Microprocessors
5. Freescale AltiVec^{TM} Documentations, “Freescale Semiconductor’s AltiVec Technology” http://www.freescale.com/files/32bit/doc/fact_sheet/ALTIVECWP.pdf

6. Apple – Power Mac G5 – Performance http://www.apple.com/lae/powermac/performance/
7. "Computer Architecture, A Quantitative Approach" by John L. Hennessy and David A. Patterson, 3rd Edition

Benchmarks

1. Veritest G5 SPEC report http://www.veritest.com/clients/reports/apple/apple_performance.pdf
2. SPEC.org cfp2000 results http://www.spec.org/cpu2000/results/cfp2000.html
3. SPEC.org cpu2000 results http://www.spec.org/cpu2000/results/cpu2000.html
4. PowerPC processor tips: Improve PowerPC 970FX performance http://www-128.ibm.com/developerworks/power/library/pa-nl17-tip.html
5. Apple – Power Mac G5 – Performance http://www.apple.com/lae/powermac/performance/
6. Fluid Dynamics Benchmark – PowerMac G5 vs G4 vs Itanium by Sean Garrick http://www.xlr8yourmac.com/G5/G5_fluid_dynamics_bench/G5_fluid_dynamics_bench.html
7. Amber 8 Benchmarks http://amber.scripps.edu/amber8.bench1.html
8. Jac/gb_mb benchmark on AMD Opteron http://apple.sysbio.info/~mjhsieh/archives/000295.html
9. Jac benchmark again http://apple.sysbio.info/~mjhsieh/archives/000334.html
10. Apple/Genentech BLAST for PowerMac G4/G5 Performance Data http://images.apple.com/acg/pdf/AGBLAST229-PerfData22Jun04.pdf
11. Spl's soapbox – apple powermac G5 http://spl.haxial.net/apple-powermac-G5/
12. Power Mac G5 http://www.apple.com/powermac/architecture.html

"Power Everywhere"

1. Power Everywhere Forum 2005 http://ascii24.com/news/i/topi/article/2005/07/07/656844-000.html
2. POWER to the people by Nora Mikes http://www-128.ibm.com/developerworks/library/l-powhist/

3. http://www.realworldtech.com/page.cfm?ArticleID=RWT072405191325
4. Mars Pathfinder FAQ http://mpfwww.jpl.nasa.gov/MPF/mpf/faqs_general.html
5. Top500 list http://www.top500.org/
6. IBM eServer BladeCenter http://www-03.ibm.com/servers/eserver/bladecenter/js20/more_info.html
7. Apple PowerMac G5 http://www.apple.com/powermac/