THE GREAT RISC-CISC DEBATE - EXECUTIVE SUMMARY

DOES RISC MEAN HIGHER PERFORMANCE?

Yes: Reduced Instruction Set Computer (RISC) designers claim to achieve two to three times the system performance of Complex Instruction Set Computer (CISC) designs at the same clock rate. Performance gains are achieved via pipelining, a well-known form of parallelism, and optimizing compilers. Deep pipelining is easier to implement in RISCs than it is in CISCs, because RISC instructions are designed to be of consistent length and to have consistent execution times. RISC architecture, constant instruction length, and constant execution time also make it easier to optimize the compiler—a critical determinant of a computer system's speed.

No: CISC designers respond that they are carefully evaluating their software base to determine the most commonly executed instructions and then optimizing the microcode or hardwiring the instructions so that they execute very quickly. CISC advocates believe they can achieve average clock cycles per instruction close to those of RISC designs. That means comparable performance without sacrificing compatibility with the huge personal computer software base. (Little general-purpose PC software is available for RISCs.)

In addition, RISCs' large register sets hamper their performance in the emerging single-user multitasking computing environment, because the contents of the register set must be saved when the user switches between tasks.

DOES RISC MEAN A COST/PERFORMANCE ADVANTAGE?

Yes: RISC designers assert that the silicon die area for a RISC processor is much smaller because the control unit is hardwired and because the instructions are simple (less space required for instruction decode). Since complexity moves from the processor to the compiler, RISC design involves a one-time compiler engineering cost rather than a silicon manufacturing cost.

No: RISCs still have not hit the price points commonly associated with PCs, because RISC solutions are less integrated and more difficult to design.

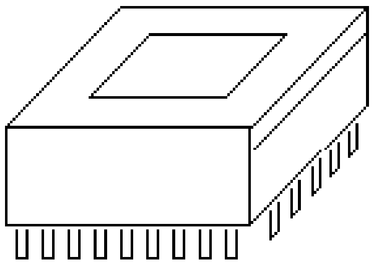
A major portion of the cost for a RISC is in the fast memory subsystem it requires. RISCs need large, fast, and sophisticated caches. Current semiconductor processes cannot pack an entire RISC processor and its memory subsystem on one die. As a result, most complete RISC designs involve several chips, which raises their cost.

Additionally, RISCs require more RAM and disk space. This code expansion is due to RISCs' reliance on many small instructions. A typical RISC machine running UNIX requires approximately 8 MB of RAM.

APPLE'S POSITION ON RISC?

The Motorola 68000 family has an impressive history of incorporating state of the art architectures and staying on the leading edge of the industry's performance offerings. We expect the 68xxx family to remain viable for years to come. We are however, always interested in exploring new architectures, including RISC. But we must remember that customers simply want a better personal computer. Implementing new technology for technology's sake does not result in compelling products. The real question is not "RISC or CISC?" but "What is the next revolution in PCs?" Until then the RISC vs. CISC debate lacks a framework within which to be discussed.

THE GREAT RISC-CISC DEBATE



THE EVOLUTION OF RISC ARCHITECTURE

RISC is an acronym for Reduced Instruction Set Computer and stands in counterpoint to “conventional” architectures now labeled CISC, for Complex Instruction Set Computer. The Motorola 68030 and the Intel 80386 are examples of CISCs; the MIPS R2000 and Sun’s SPARC are examples of RISCs. The concept of a RISC emerged from both academia and industry, notably IBM, UC Berkeley and Stanford. As CISC became more complex due to the desire to maintain backward compatibility with previous CISC designs, enhance performance, and simplify hardware implementation and software design, a new “school” of system architecture emerged that believed this complexity had high costs and resulted in sub-optimal designs. This new school of RISC designers wanted to achieve the highest *system* level performance possible, where system included not only the the microprocessor but also the hardware system (memory, support chips), compiler and operating system. As such they began designing systems that exploited the synergies that resulted from combining a highly pipelined processor with a carefully selected and simple instruction set and an optimizing compiler. I’ll explain below how these elements are combined to yield a powerful system.

CHARACTERISTICS OF RISC

There are a number of characteristics which most, but not all, people would agree differentiate RISCs from CISCs.

Single-Cycle Execution - The goal of RISC designers is to achieve the optimal performance possible for a single execution unit, an average of one machine cycle per instruction. (Many new RISCs are touting less than one cycle per instruction, but this is only possible by combining multiple execution units into the processor.) In fact, today’s RISCs do execute close to an average of one cycle per instruction. Typical CISCs (68030, 80386) execute an average of 3-10 machine cycles per instruction, though this is being rapidly reduced in the next generation of CISC microprocessors.

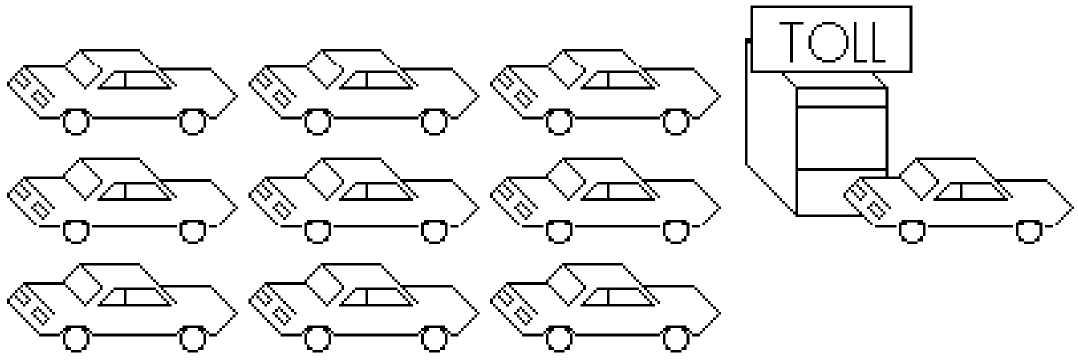
No Microcode - Microcode is a series of small operations stored within the ROM (Read Only Memory) of a microprocessor. These are used to make up the more complex instructions that the programmer is able to execute directly. (Do not repeat Judge Ingram’s famous quote from the NEC vs. Intel suit —“How would I recognize a microcode if I saw one?”—if you want to be taken seriously while discussing this topic). Rather than using microcode for control purposes, RISCs are hardwired.

Use Few and Simple Instructions, Few Addressing Modes - RISCs usually have fewer than 100 instructions, which have been designed to be executed in as predictable and consistent a number of machines cycles as possible (typically one cycle). The importance of this will be explained later. CISCs typically have 200-300 instructions, which take varying amounts of time to execute, as well as many addressing modes (5-20 vs 1-2).

Load/Store, Register to Register Architecture - In RISCs only loads and stores access memory; all the other instructions operate from register to register. This keeps the instruction execution time predictable (accesses to memory can often vary) and fast (registers are the fastest elements of a processor). In contrast, most CISC instructions can access memory. CISCs typically have from 2-16 registers; RISCs range from 32 (Motorola 88000) to 192 (AMD 29000 has the largest register file that I am aware of).

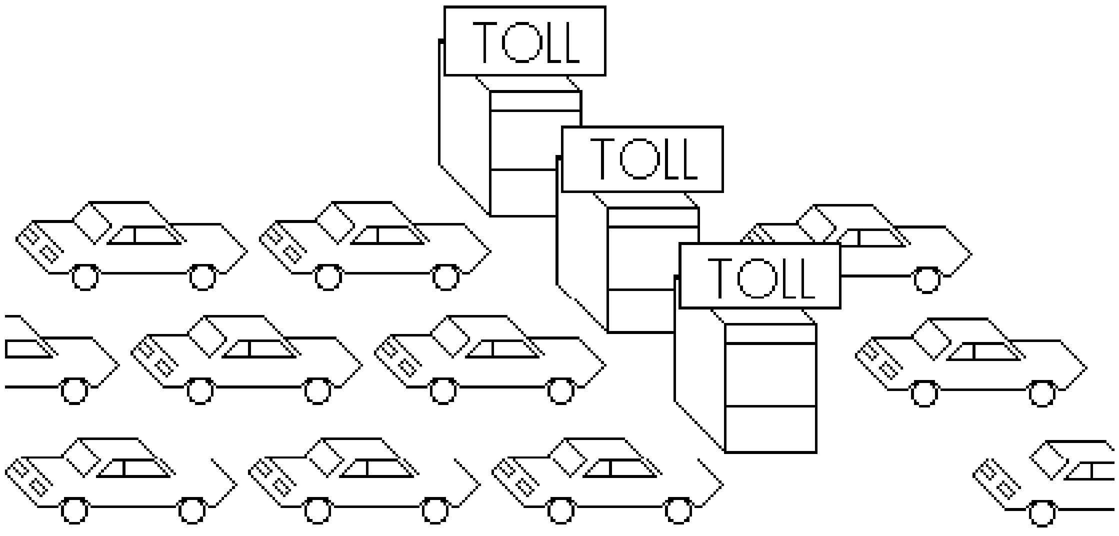
Deep Pipeline - Because of the consistency in the length and execution time of RISC instructions, RISC machines are able to take advantage of deep pipelines. This affords them a degree of parallelism that is difficult to achieve in CISCs. A very crude diagram is shown below.

Non-pipelined processor execution sequence (2 instructions in x units of time)
Fetch Decode Execute Write, Fetch Decode Execute Write



Pipelined processor execution sequence (6 instructions in x units of time)

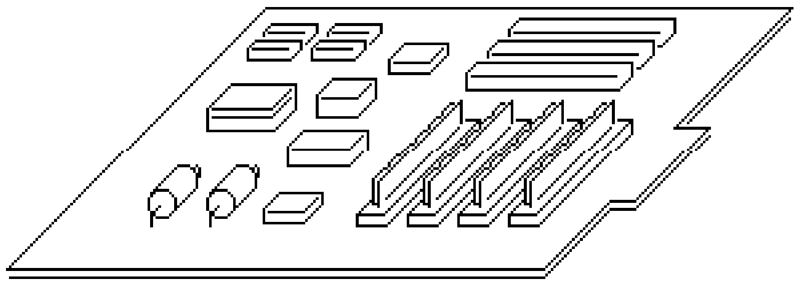
```
Fetch Decode Execute Write,
    Fetch Decode Execute Write,
        Fetch Decode Execute Write,
            Fetch Decode Execute Write,
               Fetch Decode Execute Write,
                  Fetch Decode Execute Write
```



RISC IS AN ARCHITECTURE, NOT A TECHNOLOGY - One of the common misunderstandings surrounding RISCs is the belief that they are a new technology. RISCs are a new approach to solving the classic problem confronting the computer architect—how to create a higher performance computer, more cheaply. As such, it is a new architecture, not a new technology. This statement implies that the line between CISCs and RISCs is harder to draw than it seems at first appearance. We must now deal with much more subtle issues of computer architecture rather than the more discrete issue of two separate technologies.

RISC IS BASED ON A SYSTEMS APPROACH - Many people mistakenly equate a RISC with the existence, or lack thereof, of a RISC microprocessor. A Reduced Instruction Set Computer is much more than just RISC processor. It is a carefully architected system (processor, hardware system and compiler) that has been designed to exploit the synergies between each portion of the system such that maximum performance results.

THE GOAL OF RISC IS NOT A REDUCED INSTRUCTION SET - The end result—fewer instructions—is a byproduct of the goal of designing a fast system, rather than the goal itself. To achieve the maximum synergies resulting from the interaction between the major portions of the system, the instruction set had to be of a more consistent length, had to have consistent execution times *and* had to be smaller.



THE GREAT DEBATE

Now we'll turn to the issues that are most on people's mind: How do RISCs and CISCs compare from a performance standpoint today, and what are some of the issues they will be facing tomorrow? We'll make use of the information above to delve into the issue. Performance ranking is obviously a very controversial subject where many divergent but strongly held opinions clash. There is of course no simple answer regarding whether RISCs or CISCs will win the performance race. In fact, it may be the wrong question to ask, since it is not clear that the two architectures will continue to exist in such divergent form.

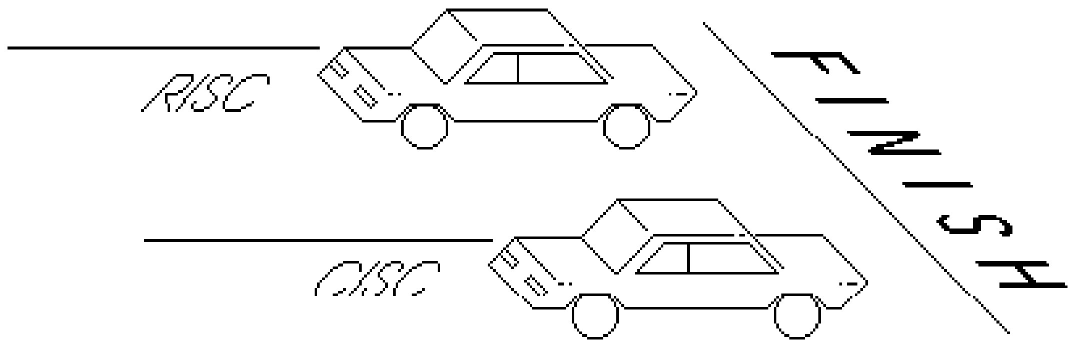
ACHIEVING SYSTEM PERFORMANCE

A good way to think about performance as it relates to the issue of RISC and CISC comparisons is to posit an equation that represents the amount of time it takes to run a given program:

$$T = C * I * CPI$$

T is the amount of time the task takes to execute and is a function of *C,* the length of the clock cycle of the processor, *I* the number of instruction it takes to complete the program and *CPI* ,the average number of clocks per instruction. In a very crude sense what RISCs attempt to do is to minimize *CPI* faster than they lose performance by the increasing number of instructions (*I*). CISCs attempt to minimize *I* by building more complex instructions, and hope that they don't lose performance as *CPI* increases. Of course, both try to minimize *C,* the clock cycle time, as it yields absolute performance increases as long as no other part of the system bottlenecks.

DOES RISC MEAN HIGHER PERFORMANCE?



RISC proponents claim 2X-3X the processor performance over CISC designs at the same clock rate. The hardware performance of a computer system can be raised in one of two ways: by speeding up the clock rate (increasing from an 8 MHz to 16MHz clock doubles performance assuming no other part of the system becomes a bottleneck) and by using parallelism (by which I mean wider data and address paths, multiple execution units, and pipelining, etc). The RISC performance gain, given the constancy of the clock rates, is achieved primarily by making use of a well-known form of parallelism, pipelining.

This is possible in RISCs to a much greater degree than is possible in CISCs because of the fundamentally different architectures. Recall that instructions in RISCs are designed to be of consistent length and consistent execution times. This makes it much easier to implement a deep pipeline in a RISC. While RISCs still face some of the same problems that CISCs face when trying to implement pipelines (pipeline stalls due to needing data that is not yet available, or conditional branches that require execution before it is clear which instruction to execute next), the level of design complexity is much less. As processor complexity increases, RISC designers believe that the CISC designers who try to include deeper levels of pipelining will be overwhelmed by the inherent complexity.

Furthermore, RISC proponents claim that their architecture favors deriving synergistic performance gains from the compiler, one of the critical components of a computer system. (The fastest hardware in the world is useless if the compilers available for it are slow.) Again, the constancy of the instruction length and execution time makes it easier to write an optimizing compiler. In fact some of the impetus for doing RISC designs

came from optimizing compiler writers who wanted an easier architecture to work with. The multiple address modes; special purpose registers; instructions of varying length and execution times; and all the variants of register to register, memory to memory and register to memory combinations make writing optimizing compilers very difficult for CISCs. The result is that most compiler writers ignore many of the complex instructions available in CISCs, and optimize their compilers for a subset of the combinations of instructions available. As a result, the hardware and software are not working as a team to achieve performance, but instead strive for performance individually, and thus less efficiently.

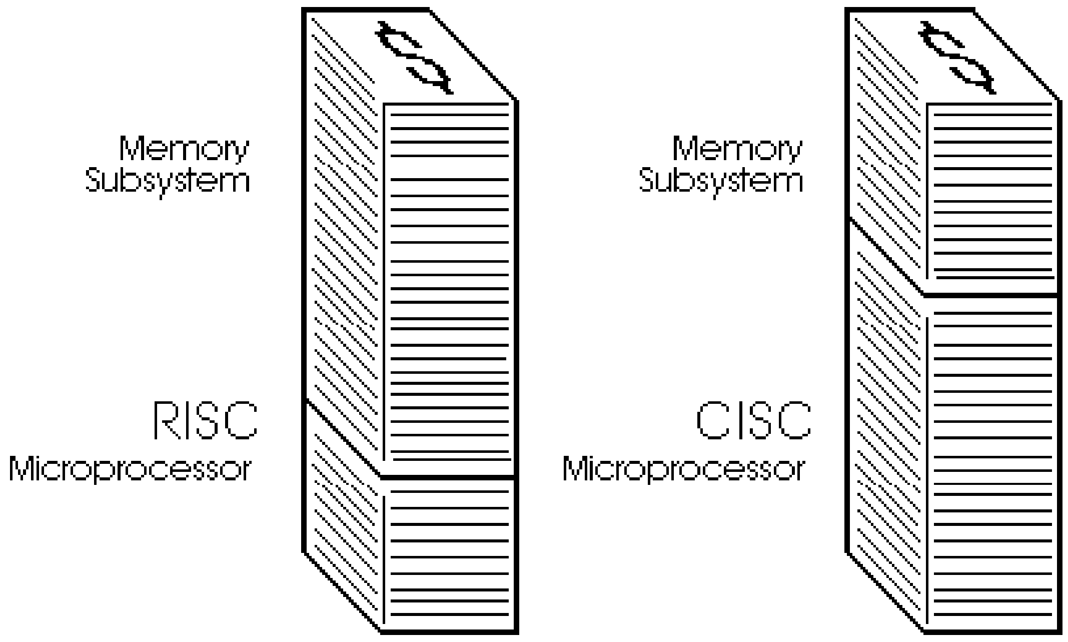
Additionally, the relative ease with which multiple execution units can be incorporated into RISCs promises to be the basis for RISCs future performance gains vis à vis CISCs. This is possible because the transistor count for a RISC execution unit is smaller because of the simplicity of the instruction decode unit.

CISC DESIGNERS' RESPONSE - Much of the performance that RISCs achieve derive from being able to take advantage of extensive and uniform pipelines. CISC designers believe that they will not be overwhelmed by the design complexity and counter that much of the complexity they are faced with is more a function of architectural baggage they must carry for software compatibility reasons than a function of the architectural constraints of CISCs themselves.

CISC proponents also point out that the large register sets incorporated in RISCs hamper their performance relative to CISCs for the emerging single-user multitasking computing environment. This is because the state of the register set has to be saved when the user switches between tasks. When working with an architecture that has a large register set, delays are encountered whenever it must be saved and restored, as the user switches between tasks.

Finally, CISC proponents emphasize that the next generation of CISC processors are primarily focused on optimizing and integrating their designs. By integrating more and more functionality (FPUs, caches) on chip they gain performance by lessening the latencies involved in going off chip. They are also carefully evaluating their software base to determine the most commonly executed instructions and then optimizing the microcode so they execute in very few machine cycles. CISC advocates believe they can achieve average clocks per instruction close to that of RISC designs. They believe that they will be able to achieve RISC-like levels of performance without sacrificing compatibility with the huge base of software that exists for PCs. If this is true it is a very powerful argument, since most software that has been developed for RISCs is engineering/scientific and little general purpose PC software exists.

DOES RISC MEAN A COST/PERFORMANCE ADVANTAGE?



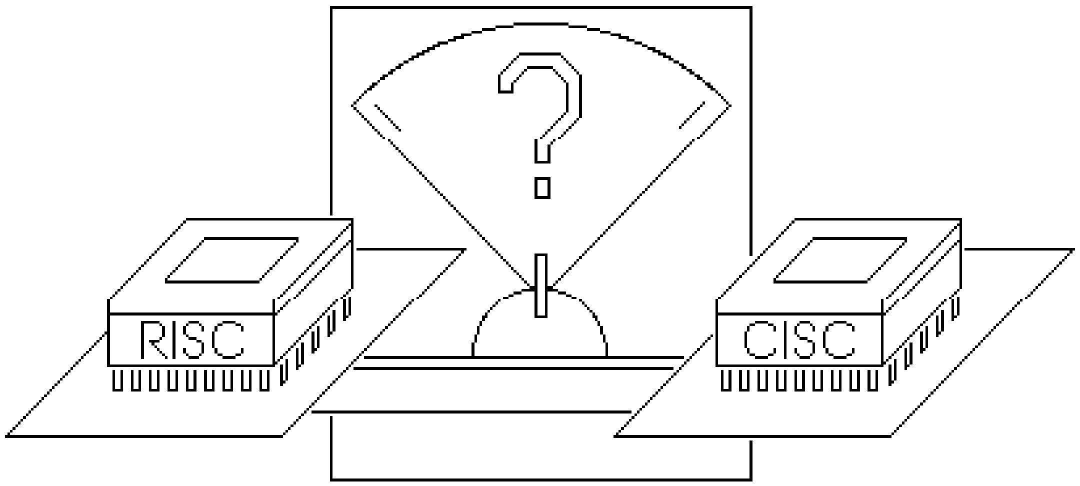
RISC proponents claim a cost/performance advantage. They say that the simplicity of their designs yields performance advantages in numerous areas. First, the silicon die area for a RISC processor is much smaller because the control unit is hardwired (no ROM for microcode) and because of the simplicity of the instructions (on many CISCs over 50% of the die area is dedicated to instruction decode, versus 10-15% for RISCs). Since some of the complexity in system design is moved from the processor to the compiler, RISC incurs a one-time engineering cost (once the compiler is written it is almost free to duplicate and ship out with each new system) rather than the large expense that CISCs incur each time they ship a large die CISC processor.

<u>CISC DESIGNERS' RESPONSE</u> - RISCs still have not hit the price points commonly associated with PCs. This is because RISC solutions are less integrated, making them more difficult to design. The lack of integration covers the range from all the small features that CISCs have acquired over the years to make them easier to design into single board PC to the more substantive issue of the complex memory subsystem needed to support a RISC processor.

A major portion of the cost for a RISC is in the fast memory subsystem it requires. Because RISCs operate at clock rates comparable to or higher than those of CISCs, and because they are averaging close to one clock per instruction, they need large, fast and sophisticated caches. Fast static caches are an expensive but vital part of a RISC design. Without them performance drops considerably due to a bottleneck at the processor/memory interface. The most commonly used "dense" semiconductor process used today for designing general purpose processors—CMOS—is not dense enough to pack an entire RISC processor and its memory subsystem on one die. As a result, most complete RISC solutions are multichip, and therefore are expensive. The ability to incorporate a RISC processor and a significant cache(s) on a single die should be achievable in the next generation of CMOS VLSI devices as lithographies approach 0.6 microns.

Additionally, RAM (and by extension, disk) requirements go up in RISCs. This code expansion is due to the increase in the instruction count that results from making use of many small instructions versus the more complex instructions used by CISCs. Depending on which RISC proponent you believe, this code expansion factor ranges from 20% to 70%. For a typical RISC machine running UNIX it is safe to assume that 8MB of RAM is the minimum requirement versus 4MB or so for a CISC machine.

THE JURY IS OUT—DELIBERATION IS UNDERWAY



As you ponder the relative merits of each of the positions, there are a number of interesting issues you should consider.

MINIMUM 2X PERFORMANCE BOOST IS REQUIRED

Historically, the PC industry has required that a new hardware platform have a minimum 2X performance superiority before significant amounts of software are ported to it. If that performance differential is surpassed, then third-party developers are forced to move to the new platform in order to be on the hardware that best showcases their software.

So questions remain: Can RISCs maintain the performance edge they have today, and can they migrate it down to PC price points? Or will the architectural baggage that will burden RISCs as they enter the PC marketplace (and are forced to maintain binary level software compatibility) slow down their performance gains? Will CISCs be able to absorb enough RISC-like architectural features (such as decreased average number of clocks per instruction) to achieve the performance boost they need to remain competitive with RISCs? If CISCs can keep within arm’s reach of RISC performance levels, CISCs’ huge installed base will be difficult for RISCs to overcome.

Or, are the RISC folks correct? Does a “pure” implementation of their architecture have an inherent performance advantage that the CISC folks cannot attain by “hybridization”? In other words, does the hybrid approach simply continue the trend that RISC people say is choking CISCs (increasing complexity that has led to performance degradation and design challenges that may eventually overwhelm even the best circuit designers)?

INTEGRATION FAVORS WHOM?

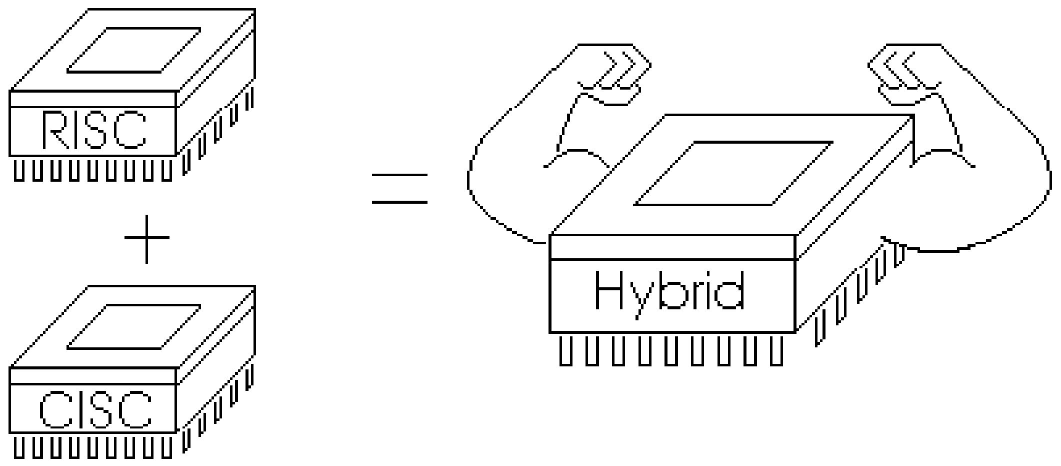
RISC advocates believe that one of the primary problems they face today— memory cost and design complexity (multichip solutions and difficult memory system designs)—will be overcome as they are able to pack processor logic and large caches on one chip. Both LSI and Cypress have full custom implementations of SPARC in design now. These efforts are being led by some of the key players involved in the design of Motorola’s 88000 and Intel’s 80486. Intel’s i860 already demonstrates the possibility for designing a large (1.2 million transistors) single chip RISC processor.

Of course CISC advocates believe that the ability to pack more transistors on a die will play into their hands. They see the additional die area as allowing them to continue to optimize their designs to run their installed base of software at performance levels near those of RISCs. They believe that the additional die area can be used to house the complex circuit designs they need to achieve RISC performance levels as well as to integrate functions that have historically been off chip. In other words, if RISC can secure performance gains by bringing caches on chip, CISC can too.

DOES IT MATTER WHAT YOU CALL IT?

I cannot close this section without adding that there is group of technical analysts who expect that this whole debate will become fuzzier rather than clearer. They expect that each of the architectures will adopt some of the features of the other (CISCs will achieve lower average clocks per instruction and will hardcode some instructions; RISCs will begin using more complex means of avoiding pipeline “bubbles” on conditional branches, such as dual prefetch and decode paths) and the distinction between the two will become less clear. For example, the 68040 has radically reduced the average number of CPI by optimizing the microcode and includes a six-stage pipeline. These analysts expect to see more processors such as the TI 340XX graphics processor family, which is often described as having a RISC CPU core with additional hardware wrapped around it.

Of course, there are also critics of the position that the architectures should be merged. They are primarily the most dedicated adherents of either RISC or CISC. These critics point out that the majority of these hybrid architectures have been designed for specific tasks—such as graphics or embedded control applications—that lend themselves to hybrid architectures. They believe that multiple hybrid architectures for specific applications will emerge, and that the RISC and CISC architectures will continue to be very separate.



SUBCURRENTS

A number of issues raised by this whole debate could be the basis for lengthy documents in and of themselves. I'd like to mention them briefly.

UNIX - The success of RISC has been tightly bound to the success of UNIX, even though there is not any particular technical connection. This is because UNIX is a relatively portable operating system, and as a result new hardware companies pushing the latest RISCs ported UNIX to their platforms. Also, because these vendors have much less of a proprietary "lock-in" factor, they are highly motivated to stay ahead of the performance curve. This leads them to exploit any short-term performance gain they can possibly achieve. So the issue of RISC vs. CISC has also grown to encompass the issue of workstation vs. personal computer. As a result, some of the debate surrounding RISCs and CISCs have really been debates about the relative merits of UNIX-based workstations vs. personal computers. We need to watch for hidden agendas as we follow the debate.

COMPETITION - Many people are concerned that the RISC companies are playing into the hands of the large Japanese electronics companies. They argue that we play to the strengths of these Japanese firms by building systems that rely heavily on commodities like DRAMs, that have relatively simple processor logic and that have an industry standard operating system. These kinds of systems give away U.S.-based companies' advantages in installed software base, proprietary operating systems, and dominance of the CISC processor market. In other words, anything that leads to standardization and commoditization of technology invites powerful Japanese competitors. The RISC people counter that it is much safer to obsolete your own products than to risk someone else's doing it—if that happens then the beldame is really over.

APPLE's POSITION ON RISC

MOTOROLA 68xxx FAMILY IS VIABLE FOR YEARS TO COME - The Motorola 68000 family has an impressive history of incorporating state of the art architectures and staying on the leading edge of the industry's performance offerings. For example, Motorola pioneered mainstream 32 bit architectures, the Harvard architecture, a flat memory architecture, and on board data and instruction caches. These are only a small number of the innovations, there are too many to list. In the performance arena, the 68040, which Motorola has announced, will enable any system that Apple might choose to build around it to run more than 20 times faster than the Macintosh Plus. We expect that Motorola will continue their history of technical innovation and expect the 68040 to offer us the performance we need to deliver the best personal computing experience available to the marketplace. In short, from an architectural and performance standpoint we expect the 68xxx family to remain viable for years to come. We expect this statement to hold true whether the 68xxx is compared to next generation CISC or RISC processors.

WE ARE ALWAYS INVESTIGATING NEW ARCHITECTURES - Apple is a company founded on innovation, it is this innovation which has allowed us to deliver the best personal computing experience in the computer industry. We are therefore, always interested in exploring new technologies and architectures. As such we are and will continue to ***investigate*** the use of RISC architectures both as CPUs within our personal computers as well as auxiliary, coprocessor, and/or imaging engines.

WE WILL MAKE OUR DECISION BASED ON USER BENEFITS - Customer simply want a better personal computer—to them, software and hardware tools are just technology used by engineers to solve a problem. For customers, the question is not “RISC or CISC?” but "What provides the best personal computing experience?" and ultimately, “What is the next revolution in PCs?” Once we understand their current and future needs in detail, we will be able to make informed choices about what technology to apply to achieve the transformation we desire. Until then the RISC vs. CISC debate really lacks a framework within which to be discussed. As we all know, strategies which have as their foundation technology for technology's sake do not result in compelling products. Discussions derivative of these strategies serve degrade the quality of the debate.

Scott Darling
Product Manager
High Performance Macintosh Product Marketing