Running Head: ARCHITECTURE EVALUATION

MetaTech Consulting, Inc.

White Paper

Evaluation of Prominent Instruction Set Architectures: RISC versus CISC

Jim Thomas

September 13, 2003

Evaluation of Prominent Instruction Set Architectures: RISC versus CISC

The general construct known as the *von Neumann machine* is the blueprint for the majority of digital computers of today as it has been for more than fifty years. The design of these components of a *von Neumann machine*, the patterns used to interconnect them, together with the *instructions* that direct the program execution (within the CPU), are components of a computer's *architecture.* The architecture of a system determines its characteristics (e.g. capabilities and performance) as well as the complexity of producing the *programs* that it is to execute.

As invention and innovation has made available more capable hardware, computer architects have attempted to exploit it to the fullest – to ensure computers provide the greatest capabilities and performance with reasonable expense and effort. The number and complexity of the set of instructions supported by the hardware increased steadily over time. Such was the trend that dominated the field of computer architecture through the middle of the 1980's. From approximately 1980, an alternate approach began to emerge that focused on revolution of the supported instruction set rather than its continued evolution. This paper provides an assessment of two diametrically opposed computer architectures followed by some of the implications of each. It concludes by asserting how they will influence the future of computing. It is assumed that the reader has a functional understanding of the principles supporting modern computer systems architecture.

Characterization of Architectures

As advances in hardware technology continued to make available an ever increasing number of digital switches (first evident as vacuum tubes, then as transistors, and more recently as semiconductors on silicon chips), computer architects have addressed the challenge of how to best exploit the increasing density. The seemingly obvious approach was simply to increase the number and complexity of the instructions the new chip would support to the greatest extent possible. Patterson and Ditzel (1980) presented an argument for a revolutionary approach that stressed the utility of sustaining only a limited instruction set and using the available chip real estate for other uses such as memory registers that could be accessed more quickly than main memory. Their paper polarized the debate by characterizing traditional architecture as a *Complex Instruction Set Computer (CISC) Architecture* and their proposed architecture as a *Reduced Instruction Set Computer (RISC) Architecture.* Though the two architectural constructs have matured somewhat over the past three decades, they remain identifiably similar to their original characterizations. The following paragraphs distinguish between RISC and CISC as they are characterized in the literature of today. In the interest of brevity, only a limited number of key factors are presented.

Number of Available Instructions

A *compiler* is used to transform the *high level language* (HLL) statements to instructions executable by the hardware. The CISC architectural philosophy strives to provide a single machine instruction for each of the HLL statements (Mano, 1993). The RISC approach is to only provide a limited number of machine instructions. This approach requires the compiler to decompose some HLL instructions into more rudimentary instructions that are supported. Table 1 illustrates that RISC computers implement approximately only a quarter of the number of instructions used by CISC computers. Patterson and Ditzel (1980) reported measurements from an IBM compiler that only 10 instructions of approximately 200 available accounted for 80% of all executed instructions and that only 30 instructions accounted for a full 99% of executed instructions. Simply put, CISC architects are expending a terrific percentage of CPU resource for instructions that will execute very infrequently and that can be accommodated through compiler transformations.

Complexity of Instructions

As mentioned previously, it is the aim of CISC architects to accommodate every HLL statement in machine instructions. Complex CISC instructions must be decoded within the CPU prior to execution. Multiple clock-cycles are necessary to execute a complex CISC instruction as illustrated in Figure 1 (Bhandarkar and Clark, 1991) that reflects that the ratio of cycles per instruction increases as the ratio of RISC to CISC instructions increases. Additionally, hundreds of thousands of bits of memory are dedicated to *control memory* to accommodate the CISC instructions where none is needed for the RISC computer. Also, Patterson and Ditzel (1980) illustrated a case in which replacing a single complex instruction with several simple instructions could perform the function 45% faster.

Novel Use of Registers

Coupled with the notion of using reduced count and complexity of instructions for the new architecture was a novel approach to the use of registers. Registers, due in part to their collocation on the CPU chip, are much quicker to access than main memory. Computers optimize the use of this resource by transferring their contents to and from main memory as portions of code are moved in and out of scope. RISC architectures employ far more registers than do CISC architecture. By using a greater number of registers, the frequency of exchanges to and from main memory are reduced. Furthermore, the registers are structured such that adjoining registers overlap to allow processes in adjacent registers to pass parameters by passing only the address (Figure 2). This is an efficiency issue not available to CISC computers through the mid 1990's.

Architectural Implications

Through layers of abstraction more simplistic views (i.e. data types, operations, and features) of the system are exposed to interface and manipulate the system (Tanenbaum, 1999). Views appropriate for those needing to interface with the hardware can be tailored for each class of user (i.e. programmer, user, other components, etc.).

An appropriate level of abstraction is characterized as exposing only those details necessary for the user to perform necessary tasks. Additionally, the abstracted view must expose the underlying hardware in such a way that the user can interface with it in a meaningful way that is sufficiently easy. An interface that is too cumbersome or complex to use will likely result in errors and unintentional features (e.g. *bugs*).

User and Software Developer

The RISC and CISC architectures have available abstractions that are suitable for the application user and software developer alike. Though there are architecture specific *software libraries* available for each class of architectures for developers, an appropriate semantic taxonomy exists for each such that they are generally equivalent to the developer.

Machine Designer Perspective

As mentioned previously in this essay, the machine designer, the architect, concerns himself with a balance between performance, cost, functionality, and complexity. Performance has been the prominent issue driving the RISC versus CISC debate. The challenge confronting architects was for many years the absence of a comparable hardware set to execute benchmark test to achieve a reasonable comparison. Bhandarkar and Clark (1991) demonstrated through execution of industry *SPEC* benchmarks on comparable hardware with similar hardware organizations. They found "from an architectural point of view (that is, neglecting cycle time), RISC as exemplified by MIPS offers a significant processor performance advantage over a VAX of comparable hardware organization" (p. 318). Findings such as these support the notion that greater performance can be achieved with a RISC architecture than a CISC architecture with a given chip space real estate.

Manufacturer Perspective

Muller and Paul (1995) suggested a formal method for measuring development factors including cost and time in respect to the fabrication of computer chipsets. They assert that the RISC design is approximately twice as cost-effective as CISC designs given a common technology and workload constraints. Furthermore, they determined that in general terms that "RISC architectures are designed for a good cost/performance, whereas CISC architectures are designed for a good performance on slow memories" (p. 6). In addition to the cost and performance of the chipsets themselves, the manufacturer must also consider the potential market for planned products prior to undertaking an effort to produce RISC chipsets. Claunch and Enck (2002) summarized the market allocation in stating that the vast majority of servers and mainframes worldwide are powered by CISC chipsets. As it requires substantial investment to migrate from one platform to another, most companies are reluctant to replace their server environments even to gain better computing performance. The maxim that the best product does not always dominate the market is equally true in the semiconductor industry as any other industry. First to market is a powerful position.

Conclusions

The RISC versus CISC debate has been waged for a little more than twenty years with advocates of both positions engaging with nearly religious vigor. As is common in philosophical debates, though it may be cast as a choice between diametrically opposed options, the most correct answer likely lies somewhere in between as a compromise. The computers of today, though they bear the brand of RISC or CISC, are in fact not pure implementations of either one – they are truly hybrid systems. RISC architects have adopted a larger set of instructions and CISC architects have realized the benefit of implementing a core set of instructions that can execute in a single CPU cycle (Tanenbaum, 1999). However, the market provides the impetus for evolution. As stated by Machanick (2001), “In practice, the biggest market is for code which is already compiled” (p. 36). This implies that the CISC-centric architectural approach which stresses the importance of evolving an instruction set that continues to support backward compatibility is likely to dominate the marketplace for the near term. This paper has, in the interest of brevity, focused on only a few of the issues germane to the CISC versus RISC debate. Interested readers are encouraged to reference current research materials available on the subject, particularly those dealing with the implications of the architecture trade-offs on the operating systems and the shortfalls that both RISC and CISC have regarding multimedia applications.

References

Patterson, D. A., Ditzel, D. R. (1980). The case for the reduced instruction set computer. *ACM SIGARCH Computer Architecture News, 8(6), 23-33.*

Bhandarkar, D. & Clark, D. W. (1991). Performance from architecture: Comparing a RISC and a CISC with similar hardware organization. *Proceedings of the fourth international conference on architectural support for programming languages and operating systems*. AMC Press, New York, NY.

Claunch, S. & Enck, J. (2002). Itanium: A qualified Success. *Gartner research notes,* SPA-17-5669.

Mano, M. M. (1993). *Computer systems architecture (3rd ed.).* Prentice-Hall, Inc., Englewood Cliffs, NJ, 07632.

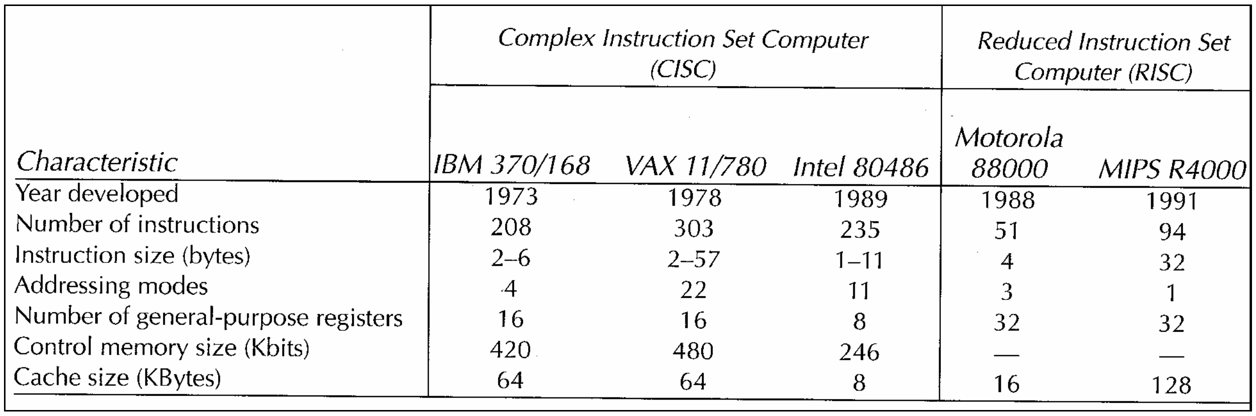
Machanick, P. (2001). Computer architecture: A qualitative overview of Hennessy and Patterson. Retrieved from http://citeseer.nj.nec.com/machanick01computer.html.

Muller, S. M. (1995). A formal method for pre-hardware cost/time trade-off analysis and selected results. Retrieved September 12, 2003 from http://citeseer.nj.nec.com/188515.html.

Stallings, W. (1996). *Computer organization and architecture: Designing for performance (4th ed.).* Prentice-Hall, Inc., Upper Saddle River, NJ, 07458.

Tanenbaum, A. S. (1999). *Structured computer organization (4th ed.).* Prentice-Hall, Inc., Upper Saddle River, NJ, 07458.

Table 1



	Complex Instruction Set Computer (CISC)			Reduced Instruction Set Computer (RISC)	
Characteristic	*IBM 370/168*	*VAX 11/780*	*Intel 80486*	*Motorola 88000*	*MIPS R4000*
Year developed	1973	1978	1989	1988	1991
Number of instructions	208	303	235	51	94
Instruction size (bytes)	2–6	2–57	1–11	4	32
Addressing modes	4	22	11	3	1
Number of general-purpose registers	16	16	8	32	32
Control memory size (Kbits)	420	480	246	—	—
Cache size (KBytes)	64	64	8	16	128

Table 1. Characteristics of Computer Architectures (derived from Stallings, 1996, p. 430)

Figure 1

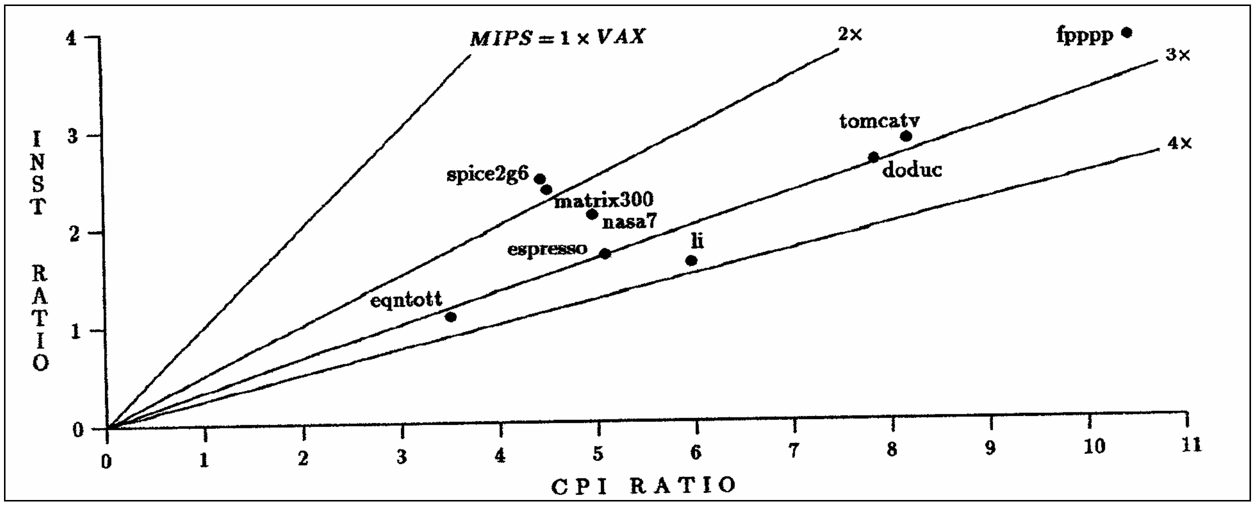


Figure 1. Instruction ratio v. cycles per instruction ratio. (Bhandarkar & Clark, 1991, p. 315)

Figure 2

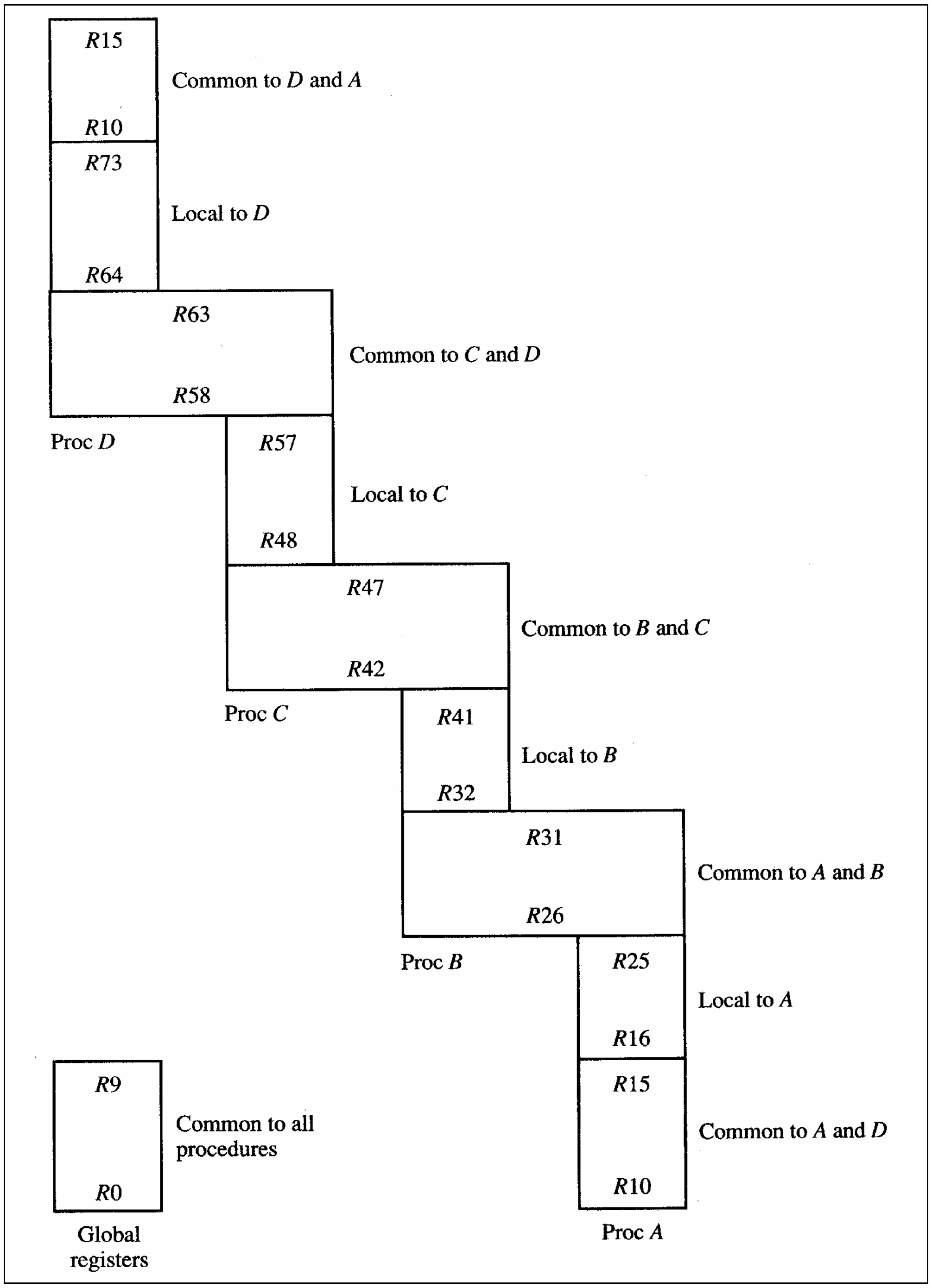


Figure 2. Input Overlapped register windows (Mano, 1993, p. 278)