RISC vs. CISC – The Post-RISC

Vasco Nuno Caio dos Santos

Departamento de Informática, Universidade do Minho
4710 – 057 Braga, Portugal
vascocaio@net.sapo.pt

Abstract. This communication analyses the birth of RISC and CISC architectures and their evolution over the past 20 years. It gives an overview of the competition between both to win the performance race, by adding new features, mainly from the opposite side and in the end converging into what is now known as Post-RISC era. The communication complements this issue by taking a brief look into novel VLIW processors, Transmeta Crusoe and Intel Itanium with the new EPIC instruction set architecture.

1 Introduction

Performance is everything. This sentence can be applied to every industry, but advances are more spectacular in the computer industry. Since the beginning of the computer era, companies have put much of their effort to make their computers faster. The primary computer component that as been target of this effort is the heart of the computer, the CPU.

Computer performance can be measured by calculating the time the computer needs to run a specific program. That equation is:

$$T_{prog} = N_{instr.} * CPI * T_{clk}$$

It can be read the following way, the time needed to run a program depends on the number of instructions (N_{instr}) t has to execute, the average number of clock cycles needed per instruction (CPI) and the clock cycle (T_{clk}).

In order to make computer chips with increased performance the terms of the equation have to be reduced to the minimum.

In the 70s the trend in developing faster computer chips was by adding more instructions, complex instructions to the instruction set. There was a reason for adding complex instructions. First, compilers were not so good, memory was still expensive, and programmer's time was expensive, and above all hardware was getting cheaper. Since programmers used High Level Languages (HLL) to make their programs and the CPU only "understood" assembly language, the translation from one language to the other was a task for the compiler.

As we can see the first term of the equation to calculate computer performance is the number of instructions the program has. If there was a specific high level instruction that was widely used and that instruction once translated into assembly would origin several instructions, why not add that instruction to the instruction set in order to get a simple correspondence? By reducing the number of assembly instructions needed to execute a program we get an increased performance.

So the main target in the 70's was to move the complexity from the software level into the hardware level and close the semantic gap between HLLs and assembly language.

In the early 80's a new trend started to emerge, and it came from IBM researchers in cooperation with the University of California Berkeley. A different approach to get increased performance appeared. The equation was still the same so, in order to get faster execution times why not try to reduce the number of cycles needed to carry out an instruction?

By analyzing how computer software was evolving and which instructions were more used, researchers found out that only about 20% of all instructions were widely used and the remaining 80% were rarely used. Their conclusion was to remove those that were rarely used, and by removing them we could obtain a simpler chip, and then try to make those 20% of instruction run faster. Their mind had a goal, "make the common case fast".

Instead of having instructions that were executed in several cycles, the main goal of this new trend was to execute one instruction per cycle. This was the optimal solution. If a complex instruction could be removed and the same task performed by several smaller instructions then there was no need for that complex instruction in the instruction set. In addition, by having simple single cycled instruction it was possible to had pipeline to these processors.

There were several restrictions added to these processors in order to achieve a simpler and faster execution of the instruction set, the elimination of the complex addressing modes, only LOADs and STOREs could access memory, there were only register-to-register operations. As a consequence there was an increase in the number of general architectural registers,.

This new approach to a different implementation of the same ISA was called RISC (Reduced Instruction Set Architecture) and a new term was born to identify the old implementation, CISC (Complex Instruction Set Architecture).

Now we have two trends, each one with their advantages and disadvantages. These two different approaches to an ISA implementation is followed by several companies, each claiming to have the best chip. Of course there are different types of consumers, with different needs.

In the next sections an overview of the evolution and convergence of the two trends into what is now called the Post-RISC era will be done.

2 Evolving

As the first RISC implementations arrived to the marketplace the success in the computational world was great, mainly cause these new chips were cheaper and faster, specially doing complicated calculus with floating point numbers. At the domestic front these chips had success when commercialized within the Mac computers.

We have now to different kinds of computers. The PC-based industry mainly dominated by Microsoft software and Intel's CISC CPUs, and the Mac-based industry dominated by Apple with RISC CPUs, both aiming at the domestic market.

Since the domestic market is the biggest and most profitable market, the industry that would have the most success would also be the industry that could invest more in developing better CPUs.

PC-based industry had the most success not because they had the best CPU but because consumers wanted compatibility to previous owned software, so they could not change platform easily.

Mac-based industry had success in the designing world due to faster chips when working with floating point numbers, and because of the easy to use Apple Macintosh operating system.

As the PC-based industry evolved several new CPU companies appeared, AMD and Cyrix. In the beginning these new companies made cheaper low-end CPUs based on the Intel technology. Nowadays they are able to make new CPUs from scratch to compete with Intel's best chips.

The competition between several companies in the same market drove the prices down and the time to market of every new chip was reduced drastically. So the CISC based market was evolving rapidly.

On the other hand, the Macintosh market was dictated by Apple alone. Without competition the prices remained stagnant and the RISC based market was evolving at much lower pace.

We already saw that RISC chips could have a much higher clock because they were simple and smaller compared with CISC chips. But we can see now that the Intel already reached the 2GHz frequency and the fastest PowerPC chip has only 800MHz Clock. The reason might be the lack of competition and demand on the Mac-based industry.

3 Converging

In the battle for performance, chip designers started to add every little thing that could give their chip an edge over other chips.

The number one rule for a RISC implementation was to "make the common case fast", but as time goes by the "common case" can change. In the 90's there was an explosion in the multimedia front. Computers were no longer used just for work they were also used for entertainment, such as earring music, playing games, watching videos, etc. Even work was no longer done in a character-oriented operating system but on a graphical-oriented operating system.

One problem aroused, the common CPU was not prepared for this kind of use. With this in mind companies like Intel added to their chip's instruction set more complex instructions dedicated to the multimedia support. Other companies added similar instructions, like instructions aimed to give better support to 3D software and games.

While adding new instructions to a CISC chip was something that we can consider normal, adding similar instructions to a RISC chip goes against the rule that these chips should have a reduced and simpler set. The PowerPC 601, for example, supports more instructions than the Pentium, and the PowerPC is considered a RISC chip and the Pentium a CISC one.

In the battle for performance, CISC chips evolved by adding features that were associated with RISC processors.

On the 486 chip, Intel started to use a 5-stage pipeline, a technique common to RISC chips. On the Pentium chip, Intel started to use branch-prediction, a super scalar architecture with 2 execution units and speculative execution.

There was still one problem to be solved in order to get all this new features to work smoothly: the Pentium instructions were still too complex.

In order to solve this problem without breaking the x86 compatibility, Intel, once again, got inspired by the RISC simplicity. So they started to have in the Pentium Pro a decoder that could separate the complex instructions into simpler ones. These simpler instructions were called micro-operations, and were then sent to the CPU. These micro-operations could get better advantages of all the new mechanisms the CPU had, like pipelines, since those instructions were smaller, the risk of having data dependencies was greatly reduced.

While we can still call them CISC chips, the Pentium Pro and newer Pentium based chips, have a RISC core inside.

As seen before, RISC chips were less hit by evolution, they already had most of the good techniques. The only difference now is that the instruction set is no longer reduced. It can be a big and complex instruction set.

But it still has a uniform instruction length, more general purpose registers, no complex addressing modes, only register-to-register operations, and load/store architecture.

These evolutions in both RISC chips and CISC chips led some groups to argue that we are now in a “Post-RISC” era.

4 The Post-RISC

Superscalar RISC processors relied on the compiler to order instructions for maximum performance and hardware checked the legality of multiple simultaneous instruction issue. Post-RISC processors are much more aggressive at issuing instructions using hardware to dynamically perform the instruction reordering. The new processors find more parallelism by executing instructions out of program order.

Figure 1 shows the flow of processing in a generic Post-RISC processor.

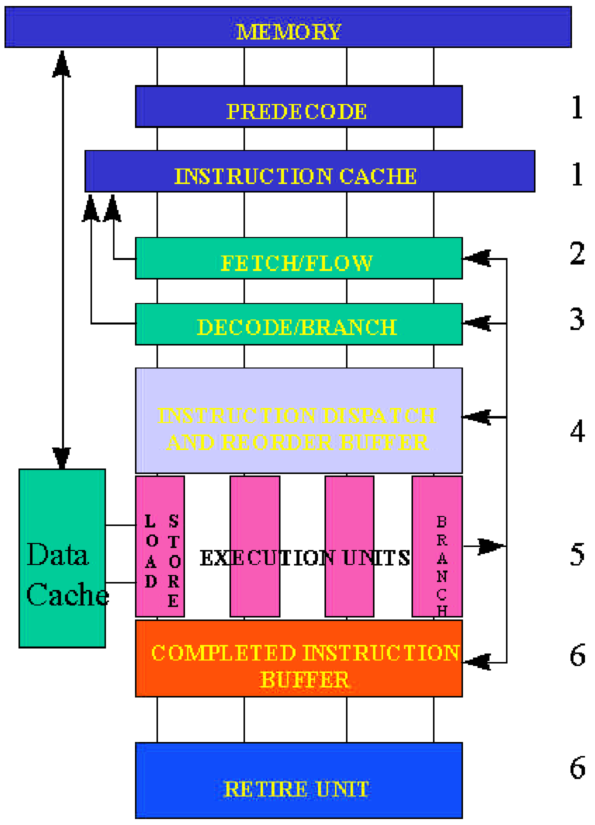

Figure 1 - The Post-RISC Architecture

Predecode occurs as the instructions are fetched from memory in groups of four. There are a few additional bits that are added to the instruction, and then the instruction is stored in the Instruction Cache. These bits (1) identify the instruction as a branch or not, (2) indicate the type of execution unit needed, and (3) determine whether or not the instruction will make a memory reference. Predecode provides a performance advantage by reducing the logical complexity of the regular decode stage and by supplying information about the instruction to stages before the decode stage.

Deciding which instructions to fetch from the I-cache is an educated guess and a wrong guess is costly. In the Post-RISC processor, branch prediction does not occur early enough in the pipeline to be used to predict the next fetch. Typically, the branch prediction is done as part of the decode operation.

At the decode/branch level (4), the instruction is decoded and more accurate branch prediction is performed. At each branch instruction, the flow of the program diverges into two separate instruction streams. Until the branch instruction is resolved, the actual instruction stream to be executed is unknown. Rather than wait idly, the CPU makes a prediction based upon certain heuristics and the past behaviour of that instruction.

In the instruction dispatch and reorder buffer, the instructions are queued waiting to be dispatched. Instructions are ready to be dispatched when their input values are available, an output register is available, and an execution unit is available. Older instructions and load instructions are given priority. Once instructions enter the buffer, program order is a secondary concern.

Execution units in the generic Post-RISC processor are similar to RISC. As in RISC, most integer execution units complete in a single cycle. Units with latency greater than one cycle are pipelined. Thus, every unit is capable of accepting a new operation every cycle. Some execution units are multi-cycle and not pipelined such as division or block-move.

The Completed Instruction Buffer holds instructions that have been speculatively executed. Associated with each executed instruction in the buffer are its results in rename registers and any exception flags. The retire unit removes these executed instructions from the buffer in program order at a rate of up to four instructions per cycle. The retire unit updates the architected registers with the computed results from the rename registers.

The Post-RISC Pipeline consists of three connected components:

(1) Fetch/Decode section, (2) Execution Units, and (3) Retire Units. Between each of these components, there is a flexible queue of instructions. The Instruction Reorder Buffer connects the Fetch/Decode components and the Execution Units. The Completed Instruction Buffer connects the Execution units to the Retire Unit.

A pipeline diagram (Figure 2) for the Post-RISC Architecture.

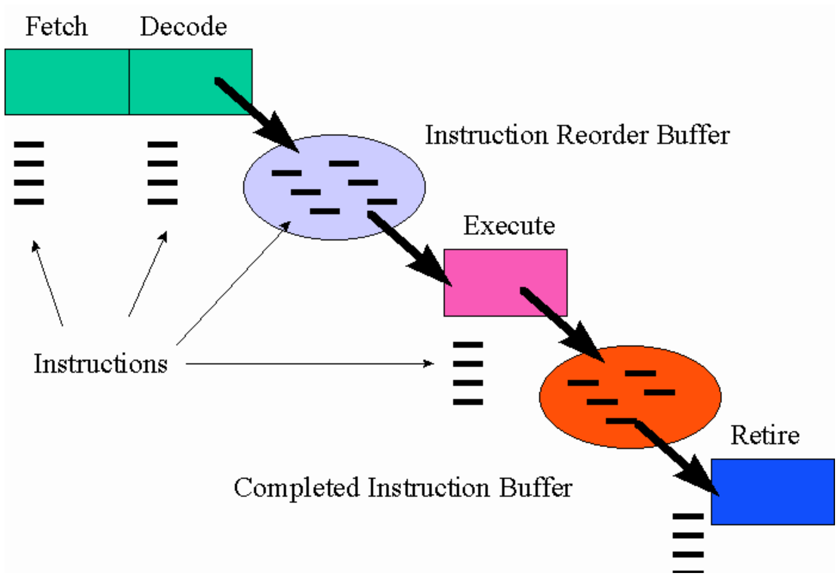

Figure 2 - Post-RISC Pipeline

The architectures of modern processors are moving toward a fundamental change in approach that we have called the Post-RISC architecture. In order to utilize increasing numbers of high-speed functional units, the processors must optimize instruction schedules at run-time. Dynamic, out-of-order scheduling is a promising technique to keep functional units busy and it is being used in many new processors.

These Post-RISC processors reduce the growing disparity between processor and memory speeds. The combination of rescheduled instructions with buffered loads and stores allows some memory latency to be hidden. However, the fundamental problem of growing memory latency with respect to processor speed remains.

5 Latest developments

5.1 Itanium

One of the latest developments, and much anticipated by the computer industry is the new Intel processor, the Itanium. The Itanium uses an entirely new form of instruction set called EPIC (Explicit Parallel Instruction Computing).

EPIC processors are capable of addressing 64-bit memory space while the x86 processors family could only address a 32-bit space, or 4GB of memory, and this was beginning to be a bottleneck in some systems.

The EPIC architecture uses complex instruction wording that, in addition to the basic instruction, contains information on how to run the instruction in parallel with other instructions.

EPIC instructions are put together by the compiler in groups of three called a “bundle”. These groups are sent to an “instruction group” with other instructions. These instructions are actually grouped and bundled together when the software is compiled. In addition the compiler handles several other important tasks that improve efficiency, parallelism and speed. The compiler adds branch hints, register stack and rotation, data and control speculation, and memory hints into EPIC instructions. The compiler also uses predication.

At the heart of the Itanium lies a parallel execution core; this core is able to run multiple instructions and operations at once in parallel. The Itanium is also deep, with a ten-stage pipeline. The first generation Itanium processor will be able to issue up to six EPIC instructions in parallel every clock cycle.

5.2 Crusoe

One of the latest developments that has the potential to revolutionize the computer industry is the appearance of a new processor called Crusoe. Transmeta, a recent company in the computer industry, developed this new processor with several goals in mind.

The primary goal was efficiency. And by efficiency they meant the lowest possible power consumption, and a level of x86 application performance that provided a reasonably good user experience.

So in order to achieve these goals, computer architects had to start the conception of the processor from scratch and question every feature other companies had in their processors. The original idea that gave birth to the RISC era was taken a step further. Every feature that the common processor had was questioned in order to see if its importance was really necessary. By reducing the hardware complexity and increasing the responsibility of compilers in the optimization of software, the size of the processor diminished severely thus reducing power consumption and production cost.

The Crusoe is now known as a “hybrid software-hardware” CPU. Many of the functions that normal x86 CPUs do in hardware, Crusoe does with a complex software layer called Code Morphing Software. [8]

The Code Morphing Software is stored in a special ROM, and is the first thing the machine loads on boot-up. After the Code Morphing layer is loaded, the OS and applications can be loaded on top of it.

Crusoe’s instruction format is a variation of VLIW (Very Large Instruction Word). Individual operations are called “atoms” and these “atoms” are packed together into 128 or 64 bit chunks called “molecules”. Molecules correspond to EPIC’s “bundles”. While EPIC’s bundles have 3 instructions, Crusoe’s molecules contain 4 atoms.

The Crusoe still has all the features the other processors have, but it does not need extra silicon to do it, the Code Morphing layer does it all in software, thus simplifying the processor, reducing power consumption and heat dissipation.

With all these revolutionary ideas, Transmeta niche market is the portable devices. These devices need processors that have x86 compatibility, low power consumption and low heat dissipation. All this requirements fit in the Crusoe processor. The portable devices market is undoubtly the one that has the most promising growing potential.

6 Concluding remarks

Over the last 20 years we have seen numerous changes in the computer architecture level. The CPU (Central Processing Unit) has been one of the targets of these changes in order to maximize performance.

In the late 70s two different approaches to the implementation of an Instruction Set coexisted, the CISC approach, that tried to improve performance by increasing the instruction set thus minimizing the number of instructions a program had, and the RISC approach, that by minimizing the Instruction Set forced compilers to take an active role in software optimization. This way processors chips became simpler and smaller thus faster.

Through the years the two approaches converged by incorporating in their processors features that the opponent had. In order to make their processors run faster companies had no problem in overlooking the basic principles of the original idea behind each approach.

As the difference between CISC processors and RISC processors diminished a new term was created. The Post-RISC era identified these new processors.

With the arrival of the Itanium the computer market may change drastically, the Itanium takes the ideas behind RISC a step further and that might be the end of the actual RISC processors. The CISC processors due to their legacy in the PC market might survive longer than the RISC ones.

The Crusoe for now is a processor that has been used where power consumption and heat dissipation is critical, but there are no certainties that Transmeta will not try to make versions of Crusoe that can compete in the PC market.

References

[1] Stokes, Jon: RISC vs. CISC: the Post-RISC Era. http://www.arstechnica.com/cpu/4q99/risc-cisc/rvc-1.html

[2] Morris, John: Computer Architecture – The Anatomy of Modern Processors. http://odin. ee.uwa.edu.au/~morris/CA406/CA_ToC.html. Course Notes on Computer Architecture

[3] Brehob, M., Doom, T., Enbody, R., Moore, William H., Moore, Sherry Q., Sass, R., Severance, C.: Beyond RISC – The Post-RISC Architecture, Michigan State University Department of Computer Science, Submitted to IEEE Micro May 1996 Submitted to IEEE Computer May 1997

[4] DeMone, Paul : RISC vs. CISC Still Matters.

[5] Gerritsen, Armin : CISC vs RISC. http://cpusite.examedia.nl/docs/cisc_vs_risc.html

[6] Simon, Jon: Itanium Technology Guide.http://www.sharkyextreme.com/hardware/ guides/itanium

[7] Morimoto, Carlos E.: Caracteristicas e Recursos dos processadores. http:// www.guiadohardware.net/tutoriais/processadores/caracteristicas_e_recursos/parte1.asp

[8] Stockes, Jon: Crusoe Explored. http://www.arstechnica.com/cpu/1q00/crusoe/crusoe-1.html

[9] Miranda, Vasco: Crusoe – An Approach for the New Era Computing, ICCA 2002 , Departamento de Informatica - Universidade do Minho

[10] Teixeira, Guilherme: Strategies to Improve Performance in the IA64 Architecture, ICCA 2002, Departamento de Informatica – Universidade do Minho