CMPSCI 691ST Systems **Fall 2011**

Lecture 6

Lecturer: Emery Berger *Scribe: Sean Barker*

6.1 The Case for the Reduced Instruction Set Computer

In this lecture we discussed the classic architecture question of reduced instruction set computers (RISC) versus complex instruction set computers (CISC). Generally speaking, CISC refers to an instruction set architecture (ISA) in which there are many instructions, typically including both powerful and special-purpose operations. In contrast, a RISC ISA is focused on a comparatively compact and efficient set of instructions, which can be used as building blocks for implementing more complicated operations.

There is no clear dividing line between RISC and CISC – in theory, RISC could be taken to an extreme and be comprised of a bare minimum set of instructions, but this is never done in practice. Rather, the distinction is one of guiding principles rather than quantitative differences. The Patterson paper discussed makes the argument in favor of the guiding principles of RISC.

6.2 History of the x86

The CISC x86 ISA is the most visible architecture today, but has evolved substantially from its original roots. The original 8080 and subsequent models such as the 8088 had no support for floating-point operations, which were considered specialized and not needed in general computation. Operations such as square roots invoked assembly line routines (such as Newton's method for square roots), which was considered acceptable, but was nonetheless extremely slow. Beginning with the 8087, Intel added a floating point *coprocessor* for handling FP operations.

Digression: Model Checking

Famously, the original 8087 floating point coprocessor had a division bug that sometimes gave incorrect results, which proved to be both costly and embarrassing for Intel. Intel's primary competitor AMD responded with the first formal proof of hardware correctness for their own floating point unit. Though this proof was correct, formal verification is rarely done today, since it is immensely expensive and time consuming. Instead, the modern approach is 'model checking', which broadly refers to a testing and verification methodology of hardware and software designs, and for which Ed Clarke and E. Allen Emerson eventually received the Turing Award. Though largely considered a hack by proponents of formal verification at the time, model checking is the de-facto standard used today.

Intel's FPU remained a coprocessor through many models, including the 8086, 80186, 80286, and 80386. Starting with the 80486, however, the floating point functionality from the coprocessor was absorbed into the main processor, in effect merging both instruction sets under the expanding x86 ISA. This tradition continued with the Pentium onwards. A similar effect is being witnessed today with graphics processors (GPUs). Previously kept strictly separate from the CPU, GPUs are increasingly moving onto the main processor, and some believe that GPU instructions will ultimately be rolled into the main instruction set, just as occurred with Intel's FP coprocessor.

6.3 The CISC Trend and Intel

The continuing additions of extra functionality to the ISA (which RISC proponents might call "ISA bloat") runs completely contrary to RISC ideals. One significant historical factor in this trend is that of Intel itself. It is important to remember that Intel is in the business of selling silicon, not designing the best architecture, and the x86 ISA is simply a tool to enable their manufacturing business. Intel has many of the best chip manufacturing fabs in the world, providing high yields, low defect rates, and high feature sizes, and this has enabled them to sell increasingly complex x86 chips at very high margins. Potentially, there will be a shift to the outsourcing of Intel's manufacturing capabilities to others who wish to make their own chips. This sort of arrangement would underscore Intel's primary interest as a manufacturer.

6.3.1 Backwards Compatibility and the Itanium Experiment

A negative side effect of the increasing complexity of the x86 ISA was the requirement of backwards compatibility, which meant that old or rarely used instructions could never be thrown away. In particular, one sore point in the x86 ISA was the lack of 64 bit support, which was highly desired by customers wanting to build servers with more than 4 GB (or, more typically, 2 or 3 GB) of memory. Intel's most notable initiative to break the cycle of backwards compatibility and start anew to some extent was the Itanium architecture, a joint effort with Hewlett-Packard whose primary benefit was to provide full 64-bit support, at the price of backwards compatibility. Around the same time, AMD released the x86-64 extension, which also provided 64-bit capabilities, but crucially maintained backwards compatibility with the main x86 ISA. The end result was that customers almost universally flocked to x86-64, and Itanium (sometimes derogatorily referred to as 'Itanic') was effectively a flop that failed to gain any traction.

6.3.2 The RISC Subversion

Many RISC-style competitors have come and largely gone, crushed by Intel's CISC ISA. Examples include Alpha, MIPS, and PowerPC. However, around the time of the Pentium, Intel essentially modified its chips to be RISC chips at heart, while still being capable of handling the full CISC ISA. The new chip architecture is illustrated in Figure 6.1, and broadly consists of two parts: a dynamic binary rewriter that translates CISC instructions into simpler operations called micro-operations, and a RISC core that actually executes the micro-ops. Thus, modern Intel chips act somewhat like just-in-time compilers by decomposing assembly instructions on-the-fly before execution.

This shift is largely due to the fact that it's easier to handle a small number of simple instructions than many complicated instructions, and the simpler instructions may actually be faster. An example from the x86 include replacing a zero-storage with a register XOR-ing itself to produce zero.

6.4 Benefits of the RISC Shift

There are three primary benefits that Intel captured by transitioning to a RISC core: ease of testing, flexibility, and performance.

1. **Ease of testing**. The use of micro-ops decouples the two primary tasks of the chip – translation of CISC-style instructions to micro-ops, and the execution of micro-ops in the RISC core. This significantly simplifies debugging of new designs and reduces the burden of testing execution hardware (since the RISC core's job is comparatively easy compared to the total number of instructions available).

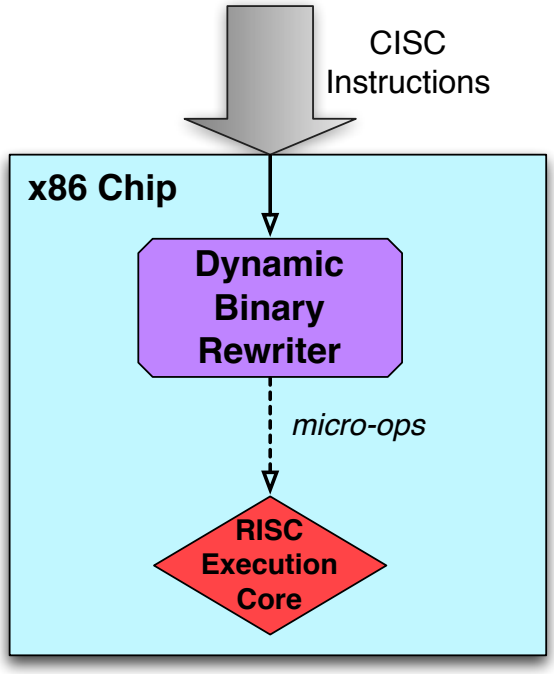

Figure 6.1: A modern CISC/RISC x86 chip.

2. **Flexibility**. Since Intel's micro-ops are not documented and are largely protected as secrets, Intel is free to change them whenever desired, even from one instance of a processor to another instance of the same processor. Since the CISC instructions are effectively an abstraction, Intel is afforded great flexibility in deciding how to implement the instruction set. Regardless of the micro-operations used, the external view remains the same.

3. **Performance**. Likely the primary reason of the RISC shift was performance. The CISC x86 instructions are both of nonuniform size and very large in some cases. The result of this is that pipelining becomes very difficult, since instructions effectively need to be decoded and pipelined in parallel. In contrast, micro-ops are compact and fixed-size, which makes capturing the speed benefits of pipelining straightforward.

Digression: x86 Density and the NX Bit

Due to the huge number of x86 instructions, the density of the ISA is very high and almost all data can be interpreted as x86 instructions. This is a significant concern from a security point of view. Traditionally, memory pages had 3 bits indicating read, write, and execute permissions respectively, and setting the execute bit appropriately would prevent unwanted code execution. However, for a long time Intel only provided 2 bits, and so the 'execute bit' was silently ignored. After AMD added this third bit, Intel finally followed suit and began advertising the feature as the 'XD bit'.

Digression: Thermal Characteristics and Dynamic Voltage
Heat dissipation is one of the most serious issues in modern chip design. Hotspots are a problem as well, since the hottest part of the chip must remain below the thermal max of the entire chip, and significant engineering efforts are used to combat this. One aspect of the multicore revolution that helps with this is dynamic voltage and frequency scaling (DVFS), which refers to underclocking a core or processor on-the-fly to reduce heat dissipation. This allows higher frequencies for temporary periods of time, after which the chip will underclock to prevent excessive heat buildup. This type of strategy is problematic for benchmarking, however, since it makes repeatable experiments more difficult.

One RISC-style argument is that compilers can more efficiently convert to RISC instructions than CISC instructions. However, this turns out not to be very important – in contrast, the cumulative compiler improvements accumulated over many years on the same ISA (x86) has resulted in quite high quality compilers for what might be considered an impenetrable ISA.

Another argument is that although CISC ISAs may have highly specialized instructions (such as crypto instructions in x86), these types of instructions only need isolated use cases to be justified for inclusion. For example, although the vast majority of x86 programs will not use crypto instructions, the few that can will be greatly enhanced by the availability of these instructions.

Digression: On-Chip Performance Counters
One of Intel's additions to assist in performance optimization has been on-chip performance counters, which provide access to various metrics such as the number of cache misses. These types of metrics provide useful insight into how the chip is actually operating on code and where extra performance can be gained. At one point, Intel was planning to remove these counters from their chips. However, Microsoft was using them to recompile code that proved to be problematic, and specifically requested that Intel not remove them from future chips. Intel agreed, and these counters are present in today's x86 chips.

6.5 Rebuttal to the RISC Argument

The second paper provided a rebuttal to many points made in the Patterson paper. One issue is the lack of comparable performance metrics – since it is difficult to have two truly comparable architectures on which to run experiments, the debate takes on a pseudo-religious tone that may rely excessively on hand-waving. A second more concrete point is the importance of code density. With a dense instruction set (CISC), a larger number of instructions will be able to fit in the instruction cache. With sparser instructions (RISC), there may be many more instruction cache misses that require RAM accesses, which are far slower than on-chip cache hits. The traditional low-level cache hierarchy is shown in Figure 6.2. The lowest-level L1 cache contains a dedicated store for instructions, while the unified L2 cache stores both data and instructions. Intel's approach essentially captures the best of both worlds: good cache behavior owing to the code density of the x86 ISA, while still maintaining the RISC benefits associated with micro-ops.

6.6 The Rise of ARM

Until several years ago, there was effectively a monoculture of x86, with most competitors effectively gone (Alpha, MIPS, PowerPC, etc). However, recently there has been a widespread emergence of the ARM architecture, particularly in mobile and low-power devices. In contrast to Intel's traditional focus on absolute performance, the primary objective in ARM is performance per watt, and ARM devices typically well outperform x86 by this metric. Though Intel is starting to compete with ARM (such as with the Atom), these efforts are hindered by several factors, such as requiring full x86 compatibility (for which energy

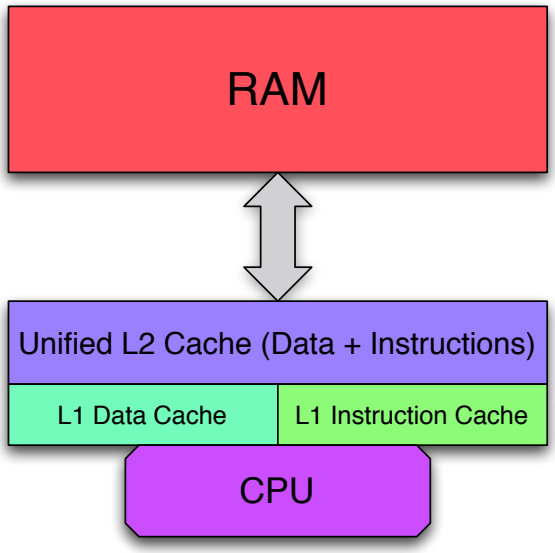

Figure 6.2: Hierarchical cache architecture.

efficiency was not a concern until recently) as well as the dynamic binary rewriter to micro-ops.

Though ARM is often called a RISC architecture and does have many RISC-style characteristics, it also has many obscure instructions focused on embedded devices, such as bit operations related to signal processing. A common but decidedly non-RISC extension to ARM is called Thumb, which is effectively a compact encoding of ARM instructions designed to improve code density (in a CISC-fashion) at the expense of increased complexity.

ARM poses a significant threat to Intel largely because the objective of the game has changed from performance to efficiency, and this has not been Intel's traditional focus. While Intel is unmatched in raw performance, this comes at the cost of size and heat, and ARM architectures look very attractive in environments for which neither size nor heat are well-tolerated, such as in mobile devices.