CMPSCI 691ST Software Systems **Fall 2011**

Lecture 5

Lecturer: Emery Berger *Scribe: John Vilk*

5.1 Architecture of the IBM System/360

For this class, we discussed the paper “Architecture of the IBM System/360” and segued into discussing various topics related to system architecture.

5.2 Loose Coupling of Hardware Implementation and Interface

Previous to the System/360 architecture, it was common for systems to have a tight coupling between their hardware and interface. Essentially, the features of the hardware would lead to the design of the interface. For the System/360, the instruction set architecture was loosely coupled with the micro-architecture such that programs correctly designed for the ISA are portable among different System/360 models. This is an instance of separating “policy” from “mechanism” or “what” from “how”.

Today we have a bit of a monoculture around x86, although ARM is gaining traction (side note: Emery strongly dislikes ARM). Much like the System/360, programs written for the x86 instruction set are portable to other x86 processors. The effect is that programs written for the Intel 8088 are technically still compatible with modern x86 processors (ignoring issues regarding OS, etc). The same is true for the System/360: programs written for it still run today on IBM’s System Z mainframes, illustrating the long life of this architecture.

5.3 Virtualization

Virtualization started with the System/360. This is an area where the x86 architecture can be seen as backwards looking compared with the older System/360 architecture. It is very difficult to achieve hardware supported virtualization with the x86 instruction set, although it has just recently started landing in chips today (side note: supposedly, its performance is not that great, though).

Virtualization gives the impression that a program is running on one machine when it is actually running on another. Our discussion during this class centered around virtualizing a machine with the same architecture as the host machine. In order to do this, a few problems have to be solved, since the virtual machine must be **sandboxed** such that the programs it runs are isolated from the host OS:

- The virtual machine’s address space must be virtualized.
- Processors nowadays typically have a user mode, for regular programs, and a protected mode, for the OS, and that’s it. The VM must not be able to use protected mode to access or mess with I/O devices.
  - As a side node to this, not all architectures only have two modes. The Multex architecture had “rings” of modes.

The first bullet has been present in the x86 for a long time through hardware memory protection, although it took awhile before OSs started using it (side note: Windows did not use it until the NT kernel).

The second bullet is a huge issue, since the VM's OS will try to make protected system calls. The only way to allow this safely without hardware virtualization support is to intercept every instruction from the VM to ensure it is translated properly to a safe instruction. This is quite slow, although VMWare managed to get around this by acting as a JIT compiler.

One of the reasons why x86 is so popular today despite having a number of shortcomings is due to "worse is better". Oftentimes shipping a working product is more important than spending a long time trying to make something perfect (scribe note: see GNU Hurd). One example of this is the famous competition between VHS and Betamax. While Betamax tapes offered superior picture quality, the format had to be licensed from Sony, and so they cost a bit more than VHS tapes. As we all know, VHS won the war, meaning the quality of the standard is not the only important factor for widespread adoption.

5.4 Stats Based Approach to Architecture

The paper justifies its choice of using base 16 floating point numbers based on circuit simplicity. The authors cite a statistical study of programs that concluded that most digit shifts can be covered with fewer circuits using this base over base 2. The choice of base 16 comes with accuracy tradeoffs. The authors did *not* have statistical data concerning the desired accuracy of floating point calculations in most programs, which led to the decision to support both 32-bit and 64-bit floating point numbers in the System/360 architecture.

The floating point decision is an early example of using statistics to guide the design of architecture. In fact, *Computer Architecture: A Quantitative Approach* by Hennessy and Patterson is a popular computer architecture textbook. This book argues that computer architecture decisions should be guided by numbers. Some people dislike this approach, but it is one way to compare architectures.

Not all architecture decisions come from a quantitative approach. In fact, some innovations come from pure insight, such as the processor cache.

5.5 Systems Research and Benchmarking: Don't Do This At Home

Most systems research is entirely benchmark driven. What is a benchmark? Literally, it is an actual mark in a bench that estimates the rough size of an object. In systems research, it is a measurement that should be representative of overall performance.

However, what is a representative program? One can make a judgment about performance of general programs based on small benchmark programs, but these results are almost never true. One classic example is the *Sieve of Eratosthenes* benchmark, which is a prime number generator. One particular company decided to hardcode its compiler to recognize this particular benchmark program, which would cause it to output hand tuned assembly code. This caused an uproar, since it was considered cheating.

Today, there is the SPEC standardized benchmark set, which is used as the "bread and butter" of architecture research. Compilers are tailored to make SPEC faster. Various knobs are tuned for the best benchmark result. Essentially, to borrow the machine learning term, **overfitting** the benchmark is a problem.

Current systems papers usually have numerous graphs. But there are no graphs in the System/360 paper. This paper is about *ideas*, not numbers. The authors use reasoning rather than numbers to convince the reader that their decisions are sound. However, today, it is still a good idea to include some sort of number to ensure that an idea works in practice. Many ideas sound great on paper, but work poorly in practice. At the same time, it is important to avoid overrelying on benchmarks. One way to do this is to use widely used programs as a benchmark (e.g. Firefox), which is what Emery typically does alongside standard SPEC benchmark results.

5.6 Next Class

Next class, we will discuss the paper on Moore's Law and Amdahl's Law. It is recommended that students look at the Wikipedia page for Amdahl's Law, since that paper does not explicitly mention what Amdahl's Law is. And it is a good idea to review what Moore's Law is too, since many people misunderstand it.