Computer Science 703

Advance Computer Architecture

2004 Semester 2

Lecture Notes
10May06
The Importance of Compilers

James Goodman

Department of Computer Science

"Compilers and Computer Architecture"

William A. Wulf, "Compilers and computer architecture," *IEEE Computer*, July 1981, pp. 41-47.

Compilers & Computer Architecture

Six costs to be considered:

- Designing (writing) compilers *(one-time cost)*
- Designing the architecture *(one-time cost, long life)*
- Designing the implementation *(one-time cost, short life)*
- Manufacturing the hardware *(only major cost reduction)*
- Executing the compiler
- Executing the compiled program

ISA Desiderata

- Regularity
  - If something is dones in one way in one place, it ought to be done the same way everywhere: "The law of least astonishment."
- Orthogonality
  - Should be possible to divide machine definition into separate concerns and define each in isolation
- Composability
  - Follows from first two: should be possible to compose orthogonal, regular notions in arbitrary ways

One vs. all

There should be precisely one way to do something, or all ways should be possible

Primitives, not Solutions

It is far better to provide good primitives from which solutions to code generation problems can be synthesized than to provide the solutions themselves

RISC vs. CISC

The Reduced Instruction Set Computer proposed by Patterson & Ditzel

Argued for

- Single-cycle operations
- Load/store design
- Hardwired control
- Relatively few instructions and addressing modes
- Fixed instructions format with consistent definition
- More compile-time effort
- Register Windows

Summary of the Controversy

- What was really happening was the discovery of pipelining and the recognition that microprogramming was not readily compatible with pipelining.
- Complexity is not inherently bad. Pipelining is more complicated to understand, but pays off big.
- Simple instructions are easier to pipeline, but complex instructions can be "cracked."
- The "Semantic Gap" is the gap between a high-level concept and assembly language instructions to implement it. It had been argued that this gap should be closed. The VAX attempted to close it by implementing single-instruction, microcoded sequences for procedure call, loop control, and interrupt handling.
- The instruction set should focus on performance, and not try to implement language-specific concepts.

"Good" RISC ideas

- Fixed length instructions (or at the least, a small number of lengths)
- Fixed instruction format with consistent use
- Primary use of Load/Store instructions for accessing memory
- Hardwired control (except for compatibility constraints)
- Depending on the compiler to close the semantic gap

“Bad” RISC ideas

- Instructions should be simple
- Small number of instructions
- Small number of addressing modes
- Single-cycle operation
- Multiple register sets: windows

“Good” CISC ideas

- Multiple sets of registers
- Architecture independent of implementation

“Bad” CISC ideas

- "Bad" CISC ideas
  - complex, language-dependent instructions
  - many, but not all, addressing modes