CS1101: Lecture 29

The Instruction Set Architecture Level

Dr. Barry O'Sullivan
`b.osullivan@cs.ucc.ie`

Course Homepage
`http://www.cs.ucc.ie/~osullb/cs1101`

Lecture Outline

- The Context
- The Role of the ISA Level
- ISA & Backward Compatibility
- What makes a good ISA?
- Properties of the ISA Level
- Memory Models
- Registers
- Instructions
- **Reading**: Tanenbaum, Chapter 5, Section 1

The Context

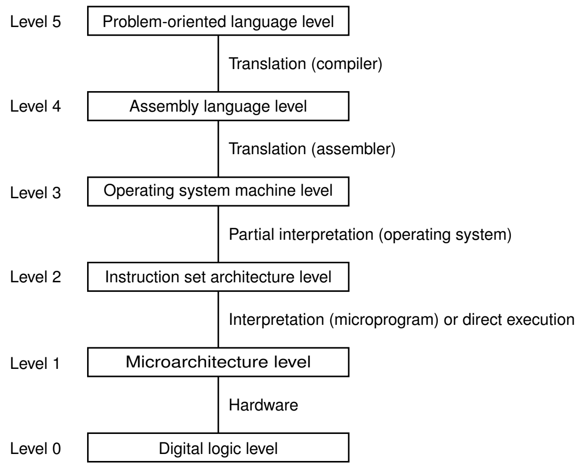
Level 5
Problem-oriented language level
Translation (compiler)
Level 4
Assembly language level
Translation (assembler)
Level 3
Operating system machine level
Partial interpretation (operating system)
Level 2
Instruction set architecture level
Interpretation (microprogram) or direct execution
Level 1
Microarchitecture level
Hardware
Level 0
Digital logic level

Figure 1.2. A six-level machine.

Introduction

- The Instruction Set Architecture (ISA) level is positioned between the microarchitecture level and the operating system level
- Historically, this level was developed before any of the other levels, and, in fact, was originally the only level.
- It is not unusual to hear this level referred to simply as *"the architecture"* of a machine or sometimes (incorrectly) as "assembly language."
- The ISA level has a special significance that makes it important for system architects: it is the interface between the software and the hardware.

The Role of the ISA Level

- The approach that essentially all system designers take is to have programs in various high-level languages be translated to a common intermediate form – the ISA level – and build hardware that can execute ISA-level programs directly.

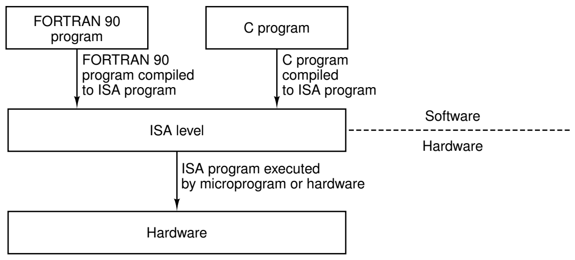
FORTRAN 90 program
C program
FORTRAN 90 program compiled to ISA program
C program compiled to ISA program
ISA level
Software
Hardware
ISA program executed by microprogram or hardware
Hardware

Figure 5-1. The ISA level is the interface between the compilers and the hardware.

ISA & Backward Compatibility

- Computer Architects are under a great deal of pressure to keep the ISA the same between models, or at least make it **backward compatible**.
- This means that the new machine must be able to run old programs without change.
- However, it is completely acceptable for the new machine to have new instructions and other features that can only be exploited by new software.
- As long as the designers make the ISA backward compatible with the previous models, they are pretty much free to do whatever they want with the hardware.
- The challenge then becomes building better machines subject to the backward compatibility constraint.

What makes a good ISA?

- A well designed ISA has significant advantages over a poor one, particularly in raw computing power versus cost.
- For otherwise equivalent designs, different ISAs might account for a difference of as much as 25
- A good ISA should define a set of instructions that can be implemented efficiently in current and future technologies, resulting in cost-effective designs over several generations.
- A good ISA should provide a clean target for compiled code.
- In short, since the ISA is the interface between the hardware and the software, it should make the hardware designers happy (easy to implement efficiently) and make the software designers happy (easy to generate good code for).

Properties of the ISA Level

- In principle, the ISA level is defined by how the machine appears to a machine language programmer.
- Let us regard ISA-level code to be what a compiler outputs.
- To produce ISA level code, the compiler writer has to know:
  - what the memory model is;
  - what registers there are;
  - what data types and instructions are available, and so on.
- The collection of all this information is what defines the ISA level.

Memory Models

- All computers divide memory up into cells that have consecutive addresses.
- The most common cell size at the moment is 8 bits – called a byte.
- The reason for using 8-bit bytes is that ASCII characters are 7 bits, so one ASCII character plus a parity bit fits into a byte.
- If UNICODE comes to dominate the industry in the future, then future computers may be based on 16-bit consecutively numbered units.

Registers

- All computers have some registers visible at the ISA level.
- They are there to control execution of the program, hold temporary results, and for other purposes.
- In general, the registers visible at the microarchitecture level are not visible at the ISA level.
- However, a few of them, such as the program counter and stack pointer, are visible at both levels.
- On the other hand, registers visible at the ISA level are always visible at the microarchitecture level since that is where they are implemented.

Instructions

- The main feature of the ISA level is its set of machine instructions.
- These control what the machine can do.
- There are always LOAD and STORE instructions (in One form or another) for moving data between memory and registers and MOVE instructions for copying data among the registers.
- Arithmetic instructions are always present, as are Boolean instructions and instructions for comparing data items and branching on the results.