CO113 - Architecture

Prelude

The content discussed here is part of CO113 - Architecture (Computing MEng); taught by Wayne Luk, and Jana Giceva, in Imperial College London during the academic year 2018/19. The notes are written for my personal use, and have no guarantee of being correct (although I hope it is, for my own sake). This should be used in conjunction with the lecture slides, *The Hardware/Software Interface Class by Luis Ceze and Gaetano Borriello* on YouTube, and *Computer Organization and Design : The Hardware / Software Interface (Fifth Edition)* (chapters 1 to 4, and appendices B, and D), by Patterson, D., and Hennessy, J.

Lecture 1

Computer architecture is a combination of ISA (instruction set architecture), and machine organisation. We can see the ISA as an interface between the high level software, and the capabilities of the physical hardware components.The benefit of having the ISA is that a piece of software can be compiled into an instruction set, and then be reused on different hardware. For example, near identical versions of the x86 instruction set are used in Intel, and AMD chips despite the two having drastically different internal designs. On the other hand, microarchitecture, or computer organisation, is the the way a given ISA is implemented in a particular processor. This comes with the additional benefit that code doesn't need to be reimplemented even if there is a drastic change in the future for the microarchitecture / machine organisation.

There are two design approaches, both of which have their benefits, and drawbacks;

- Complex Instruction Set Computers (CISC)

  The programs run on this design are closer to the high-level languages that we program in; which means that the compilers used are simpler. This is possible due to the decreasing size of transistors, and thus the increased number of gates on a chip. Programs on this instruction set tend to be smaller, as code can be represented in fewer instructions, thus saving storage.

- Reduced Instruction Set Computers (RISC)

  On the other hand, the programs running on this instruction set are closer to machine code, due to the smaller range of instructions. A more powerful, better optimised, compiler will be required. Additionally, the programs here are faster, since they have simpler instructions - but they may require more instructions to achieve what a CISC can do in one, thus there may be a trade-off. It's also easier to build a chip with less instructions, which leads to lower development costs. Due to the smaller physical size of the chips, we can not only fit multiple chips together, but also use the space for memory, since accessing memory outside of the chip is very slow (compared to the high-speed registers nearby).