GC03
Architecture & Hardware

Peter Rounce

p.rounce@cs.ucl.ac.uk

[Notes; Courtesy of Graham Knight , who taught this course for many years]

The Purpose of the Course

There are two main purposes:

- To “de-mystify” computers.
- To provide sufficient understanding of how a computer works that the rest of the course is comprehensible.
- To help you on way to becoming computer experts.

18/09/2008 GC03 Introduction 0-2

The course starts from the notion that you know almost nothing about computers. People who have been dissecting microprocessors since the age of 13 may be bored for some of the course. I am sorry about that. For the rest of you, the course is supposed to provide a step-by-step introduction to what goes on inside a computer. If this does not seem to be the case, it is my fault and not yours. IT IS UP TO YOU TO STOP ME AND DEMAND AN EXPLANATION! Do not assume that silence on the part of your colleagues signifies understanding - it may signify quite the reverse and they will be very grateful for your intervention.

One of the stated aims of this course is to provide an understanding of “how a computer works”. This could mean a variety of things. Suppose, for example, that the same ambition were applied to cars. Many people operate quite happily with a very “high-level” model of a car; its behaviour consists of the ability to go forward or backwards, to turn to the left or to the right. The controls that may be employed are principally “go” and “stop” pedals and a steering wheel. One description of “how a car works” could be at this level - explaining how accelerating fiercely whilst steering hard right is liable to cause a skid, and so on. Most people, however, would expect a “lower-level” description than this; one involving carburettors, distributors etc. with the controls being throttle openings, ignition timings etc. They probably would not appreciate an even lower level description involving complex thermodynamics and the chemistry of combustion.

To become computer experts, you need to know more than just how to use word and how to program.

Recommended Reading

"The Architecture of Computer Hardware and Systems Software (3rd Edition)", Irv Englander, Wiley, ISBN 0-471-07325-3

[Covers architecture, Operating Systems, and networks]

"Computer Organization and Design (3rd Edition)", Patterson and Hennessy, Morgan Kaufmann, 1999, ISBN 1-55860-4910-X

Based on the MIPS processor which we use in the course, but generally too detailed and not enough breadth.

NOT "Computer Architecture – A Quantitive Approach" by the same authors

"Structured Computer Organization (4th Edition)", Andrew S. Tanenbaum, Prentice-Hall, 1999, ISBN 0-13-020435-8

Nicely written general text

18/09/2008 GC03 Introduction 0-3

Similar possibilities for descriptions at different levels apply to computers, so our first problem is to decide at what level we will study "how a computer works". As an example, consider a specialised but (probably) familiar computing machine - the "word processor". A word processor can manipulate characters, words, lines of text, documents, font characteristics etc. A full description of its behaviour would consist of a list of all the commands it "understood", (add and delete text, move it around etc.), together with a description of its behaviour in response to each command. However, there is really no physical word processor. You may be aware of the wealth of different word processors that can be run on an IBM PC (MS Word, Word Perfect etc.).

This is possible because the IBM PC - like all computers - is a general purpose physical machine. By programming it correctly, the single "physical machine" can be made to behave like a whole variety of different "virtual machines" each with its own set of commands (i.e. its language) and behaviour. Here we can see that the analogy with a car is less than perfect. A car can only do one thing so the multi-level description is merely to assist our understanding - to break the description into convenient chunks. By contrast, a computer can be many things. Thus, although we can identify an "operating system" level we can implement that level in many different ways, each of which makes our computer look quite different to the level above. For example, on a PC we can run (amongst others) the Windows XP or Linux operating systems. However, a program written for Windows XP will not work with Linux even though the underlying hardware is the same. Windows XP and Linux produce two quite different virtual machines.

Virtual Machines

- Description can be at many levels
  - Word processor can manipulate characters, words, lines of text, etc.
  - Description of its behaviour - a list of all the commands it "understands"
  - There is no physical word processor
    Word processor is a "virtual machine"
  - Physical machine can support many different "virtual machines".
- Languages and levels

18/09/2008 GC03 Introduction 0-4

As computers have developed and become more complex, designers have found it convenient to build a series of virtual machines each constructed by programming the virtual machine below it. The aim of this course is to peel away the virtual machines until we get to a machine whose behaviour is fixed by the way it is physically constructed. To use the jargon words, we want to throw away the "software" and expose the "hardware". Two things should be noted:

- Computer architecture stops at a description of what the hardware does and does not consider how it does it. This latter is the subject of the hardware part of this course.
- The distinction between hardware and software is not always very clear; what is done by hardware on one computer may be done by software on another. For example, most modern computers have a piece of hardware that can divide one number by another. Early computers lacked this feature and when asked to divide had to run a little program which caused it to do repeated subtractions.

Tanenbaum identifies 6 main levels of description of computer systems. Each defines a machine (physical or virtual) in terms of a language and the set of things it can manipulate. A brief summary of Tanenbaum's model is included in the next section. There is no need to try to understand this model completely or to memorise it. We will refer to it occasionally and its meaning should become clearer as the course progresses.

Tanenbaum's Six Levels

- Level 5) Problem Oriented Language Level
- Level 4) Assembly Language Level
- Level 3) Operating System Level
- Level 2) Conventional Machine Level
- Level 1) Microprogramming Level

  (not all computers have this level – so not dealt)
- Level 0) Digital Logic Level

18/09/2008 GC03 Introduction 0-5

Level 5) Problem Oriented Language Level	We already have the word processor as an example at this level. You may also be familiar with rather more general languages such as Java, Pascal, Cobol and C++. The "things" these languages manipulate are usually "integers", "characters", "matrices" etc. Most computer users and programmers operate at this level.
Level 4) Assembly Language Level	Here, the virtual machine reflects the capabilities of the hardware much more closely. Few people write programs at this level nowadays. Sometimes it is done where speed of execution is crucially important. In this case, the (possible) inefficiencies introduced by translation from Level 5 must be avoided and optimisations based on the known capabilities of the actual hardware be introduced. (Students of Computer Architecture are often compelled to program at this level for their general enlightenment and the goodness of their souls)
Level 3) Operating System Machine Level	The operating system (Linux and Windows XP are examples) is concerned with the fair and efficient use of the computer's resources. In a system like Unix, which allows several users and programs to operate concurrently, one of the main tasks is to ensure that programs behave fairly and do not interfere with each other. In this case, the Operating System is providing several virtual machines concurrently (one for each program) even though there is just one physical machine. The Operating System Machine Level provides a special language for accessing shared resources and forces programs to use these rather than accessing the resources directly. Programmers concerned with the workings of the operating system are called "Systems Programmers".

Resources

- There will be a course web page at http://www.cs.ucl.ac.uk/staff/P.Rounce/teaching/gc03 and http://www.cs.ucl.ac.uk/staff/electran/gc03
- Web page will contain:
  - These notes in PDF
  - Information about coursework
  - Other bits and pieces
- You should be making your own notes, not just relying on mine

18/09/2008 GC03 Introduction 0-6

Level 2) Conventional Machine Level	A description at this level comes close to defining precisely what the hardware of a real machine is capable of doing. The instructions used at this level are precisely those which get stored in a computer's memory. In the very early days of computers, people wrote programs at this level. It could not have been pleasant - everything, instructions and data has to be represented as a series of numbers. (Nevertheless, even today, students of Computer Architecture are sometimes required to program at this level.)
Level 1) Microprogramming Level	At this level we are concerned with the precise sequence of events that goes on inside a computer when a Level 2 instruction is carried out. This is one place where the distinction between hardware and software becomes very blurred. This level is mainly the concern of the manufacturers of computer chips. Not all computers are programmable at this level – microprogramming brings flexibility but there is a performance cost.
Level 0) Digital Logic Level	The "things" that are of interest here are the "gates" inside computer chips which take one or more electrical signals as input, perform some simple computation and produce one or more outputs. Gates are constructed from electronic components such as transistors. This level is not programmable. Its behaviour is defined entirely by the way the gates are connected to each other.

Acknowledgements:

These slides came from Graham Knight with some adaptions by me, and Graham said that "Some of the slides in these presentation originated with Jon Crowcroft and Chris Clack. Others are taken from the slides made available by Patterson and Hennessey."