Operating System Foundations

Dr G.M. Bierman

Michaelmas Term 2001

www.cl.cam.ac.uk/Teaching/2001/OSFounds/

Today's Lecture

Today we'll cover:

- Introduction to the course
- Reading List
- Historical Perspective
  - including Von Neumann Architecture
- Languages and levels

Achtung!

This course has completely changed since last year!

It is now synchonized with the 1a Operating Systems course (Easter Term)

Course Aims

This course aims to:

- provide you with a general understanding of how a computer works,
- explain the structure and functions of an operating system,
- illustrate key operating system aspects by concrete example, and
- prepare you for future courses. . .

At the end of the course you should be able to:

- describe the fetch-execute cycle of a computer
- understand the different types of information which may be stored within a computer memory
- compare and contrast CPU scheduling algorithms
- explain the following: process, address space, file.
- distinguish paged and segmented virtual memory.
- discuss the relative merits of Unix and NT. . .

Course Outline

- Part I: Computer Organisation
  - Computer Foundations
  - Operation of a Simple Computer.
  - Input/Output.
- Part II: Operating System Functions
  - Introduction to Operating Systems.
  - Processes & Scheduling.
  - Memory Management.
  - I/O & Device Management.
  - Filing Systems.
- Part III: Case Studies
  - Unix.
  - Windows NT.

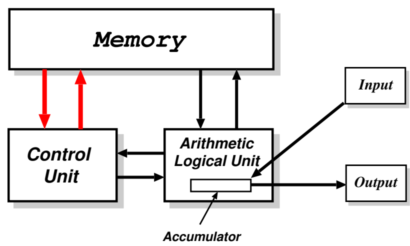
Recommended Reading

- Tannenbaum A S
  Structured Computer Organization (3rd Ed)
  Prentice-Hall 1990.
- Patterson D and Hennessy J
  Computer Organization & Design (2rd Ed)
  Morgan Kaufmann 1998.
- Bacon J M
  Concurrent Systems (2nd Ed)
  Addison Wesley 1997
  (especially Part I, and Chapters 23 & 25)
- Silberschatz A, Galvin P, and Gagne G
  Operating Systems Concepts (6th Ed.)
  Addison Wesley 2002(!)
- Leffler S J
  The Design and Implementation of the 4.3BSD UNIX Operating System.
  Addison Wesley 1989
- Solomon D
  Inside Windows NT (2nd Ed)
  Microsoft Press 1998.

A Chronology of Early Computing

- (several BC): abacus used for counting
- 1614: logarithms disovered (John Napier)
- 1622: invention of the slide rule (Robert Bissaker)
- 1642: First mechanical digital calculator (Pascal)
- Charles Babbage (U. Cambridge) invents:
  - 1812: "Difference Engine"
  - 1833: "Analytical Engine"
- 1890: First electro-mechanical punched card data-processing machine (Hollerith, later IBM)
- 1905: Vacuum tube/triode invented (De Forest)
- 1935: the relay-based *IBM 601* reaches 1 MPS.
- 1939: *ABC* — first electronic digital computer (Atanasoff & Berry, Iowa State University)
- 1941: *Z3* — first programmable computer (Zuse)
- Jan 1943: the *Harvard Mark I* (Aiken)
- Dec 1943: *Colossus* built at 'Station X', Bletchley Park (Newman & Wynn-Williams, et al).

The Von Neumann Architecture



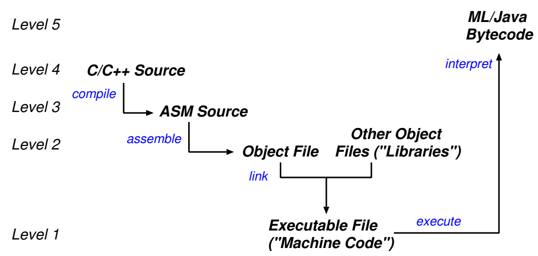
- 1945: *ENIAC* (Eckert & Mauchley, U. Penn):
  - 30 tons, 1000 square feet, 140 kW,
  - 18K vacuum tubes, 20×10-digit accumulators,
  - 100KHz, circa 300 MPS.
  - Used to calculate artillery firing tables.
  - (1946) blinking lights for the media. . .
- But: "programming" is via plugboard ⇒ v. slow.
- 1945: von Neumann drafts "EDVAC" report:
  - design for a stored-program machine
  - Eckert & Mauchley mistakenly unattributed

Further Progress. . .

- 1947: "point contact" transistor invented (Shockley, Bardeen & Brattain, Bell Labs)
- 1949: *EDSAC*, the world's first stored-program computer (Wilkes & Wheeler, U. Cambridge)
  - 3K vacuum tubes, 300 square feet, 12 kW,
  - 500KHz, circa 650 IPS, 225 MPS.
  - 1024 17-bit words of memory in mercury ultrasonic delay lines.
  - 31 word "operating system" (!)
- 1954: *TRADIC*, first electronic computer without vacuum tubes (Bell Labs)
- 1954: first silicon (junction) transistor (TI)
- 1959: first integrated circuit (Kilby & Noyce, TI)
- 1964: IBM System/360, based on ICs.
- 1971: Intel 4004, first micro-processor (Ted Hoff):
  - 2300 transistors, 60 KIPS.
- 1978: Intel 8086/8088 (used in IBM PC).
- ~1980: first VLSI chip (> 100,000 transistors)

Today: $\sim$ 40M transistors, $\sim 0.18\mu$, ~ 1.5 GHz.

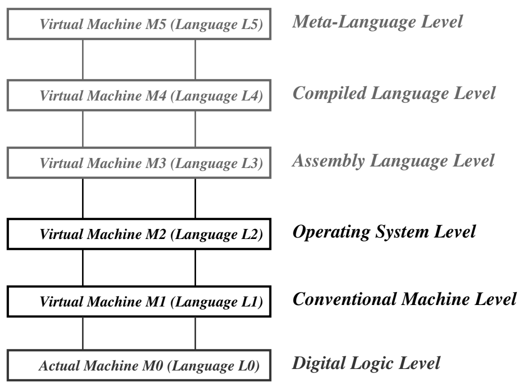
Languages and Levels



- Modern machines all programmable with a huge variety of different languages.
- e.g. ML, Java, C++, C, Python, Perl, FORTRAN, Pascal, Scheme, . . .
- We can describe the operation of a computer at a number of different *levels*; however all of these levels are *functionally equivalent*
  — i.e. can perform the same set of tasks
- Each level relates to the one below via either

  a. translation, or
  b. interpretation.

Layered Virtual Machines



- In one sense, there is a set of different machines M_0, M_1, . . . M_n, each built on top of the other.
- Can consider each machine M_i to understand only machine language L_i.
- Levels 0, -1 pot. done in Dig. Elec., Physics. . .
- This course focuses on levels 1 and 2.
- NB: all levels useful; none "the truth".

Summary

You should now understand:

- What this course is about,
- Some historical background on the material in this course,
- Von Neumann architecture, and
- Languages and levels.

Next lecture: Simple Computer Architecture I

Background Reading:

- Section 1.8 of Hennessy/Patterson
- Chapter 22 of Silberschatz et al. (OS perspective)