Computer Architecture: Introduction

Behzad Bordbar
School of Computer Science
The University of Birmingham
[Thanks to Guilin Wang and Ata Kaban]
Part 2: Module Aims

- Introduce the main hardware components of a computer and their functions.
- Explain the role of an operating system and system software.

Notes:
- This is introductory, not exhaustive!
- For further reading, refer to the following book:
- As far as this module is concerned, only the material taught in lectures is examinable! ☺
Topics Covered

- Computer architecture
  - The van Neumann computer [this lecture]
- I/O devices and networks
- Digital logic
- Microprogramming
- Machine code
- Operating system
- Compilers and interpreters
Some Historical Notes

• 1642-1945: Mechanical Computers
• 1943: 1st electronic computer (COLOSSUS), British Govt.
  - The stimulus is WW II (ENIGMA, Thomas Jefferson, Alan Turing)
• 1949: 1st stored program computer (EDSAC), Wilkes
• 1960: 1st minicomputer (PDP-1), DEC; 50 sold
• 1965: 1st mass-market (PDP-8), DEC; 50,000 sold
• 1978: 32bit (CISC) superminicomputer (VAX), DEC
• 1981: personal computer (IBM PC), IBM; millions sold
• 1987: RISC workstation (Sun), SPARC
  - RISC (Reduced Instruction Set Computer) vs. CISC.

A good history of the development of computer architectures:
Why Computer Architecture?

• People and computer are different !!!

convenient for Human

translation or interpretation

convenient for Computer

execute
Multilevel machines

- Assembly
  - translation(assembler)
- Operating System
  - translation(assembly)
  - problem oriented lang.
- Instruction set Architecture
- ISA
  - Microarchitecture
- Hardware
  - Digital logic
  - Human
    - friendly

Lecture 01
Moore’s law:

More layers, more software, then more memory and more powerful processors

Need more transistors (electronic switches) on chip.

Number of transistors double every 18 months.

see www.intel.com/research/silicon/mooreslaw.htm

notice it is not a law, an empirical observation.

... the architecture is getting more and more complicated.
The von Neumann Computer

- Designed by John von Neumann, Princeton, in 1940s.
- **Breakthrough**: digital (bits, 0 or 1) and stored program.
- Still prevailing today.
Memory

• Has **cells (words)**
• Each cell has **address** (number) and contains **n bits**

```
  513: 0100111100101000
  514: 1000101000011101
  515: 0110000000100000
...
```

• In von Neumann computer
  – 4096 word memory, 40 bit word
• Typical computer has 16 or 32 bit word and lots more memory!
Von Neumann’s Main Ideas

• Represent **data** as sequences of bits (0 or 1)
  integer 1 as 000000000001 (on 12 bits)
• Memory has 4096 cells, with 40 bits word
• Represent **instructions** as sequences of bits
  code (8 bits), **address** of argument (12 bits)
  01100000000000000001
• Store both data and program instructions in memory
  – 2 instructions per word
• Instructions
  – use argument from memory, add to **accumulator**
Simple Modern Computer

- Just two I/O devices.
- **PC** (Program Counter): address of next instruction.
- **IR** (Instruction Register): instruction being executed.

- Essentially von Neumann design.
- Difference: connected by a **bus** (parallel wires for data transfer).
- Several registers.
How It Works

- **Main memory**: stores data and program
- **Bus**: transfer of data, addresses and control signals from/to memory and devices
- **CPU** (Central Processing Unit):
  - **control unit**: fetches instructions and executes them
  - **ALU** (Arithmetic Logic Unit): performs operations (add, subtract, etc)
  - **registers** (fast memory): store temporary results and control information (address of next instruction)
- **I/O devices** (Input/Output): user communication
How ALU works

Example:
- A contains 10
- B contains 2
- ALU calculates 10+2, sends result back to register
How the Computer Executes

- Fetch instruction into IR
- Increment PC to point to next instruction
- Determine type of instruction
- If contains address, fetch contents into register
- Execute instruction
- Repeat from the above procedures
Representing Data...

- As sequences of bits
- **Characters** usually 8 bits (=1 **byte**)
  - ASCII (American Standard Code for Information Interchange) code
  - 7 bits, in total 128 characters
    - M  1001101
    - Z  1011010
    - @  1000000
    - ...
  - can be packed into words
Representing Integers

• Positive integers

\[ 150 = 1 \times 10^2 + 5 \times 10^1 + 0 \times 10^0 \]
\[ 110 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \]

• Negative integers: two’s complement, sign info on the most significant bit

<table>
<thead>
<tr>
<th>Number</th>
<th>Binary</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>1110</td>
<td>( \times(-8) + 1 \times 4 + 1 \times 2 + 0 \times 1 )</td>
</tr>
<tr>
<td>-1</td>
<td>1111</td>
<td>( \times(-8) + 1 \times 4 + 1 \times 2 + 1 \times 1 )</td>
</tr>
<tr>
<td>0</td>
<td>0000</td>
<td>( 0 \times 8 + 0 \times 4 + 0 \times 2 + 0 \times 1 )</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>( 0 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1 )</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>( 0 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1 )</td>
</tr>
</tbody>
</table>
Representing Real Numbers

• Using floating-point representation

  mantissa    exponent
  1003        -3

  \[ 1.003 \times 10^{-3} = 1.003 \]

• Can have single/double precision (stored in one word or two)
Representing Trees and Lists

- **Addresses** can be stored in memory to represent pointers

![Tree Diagram]

- 1000: 17
- 1001: **1003**  Root
- 1002: **1006**
- 1003: 13
- 1004: -1  Left Child
- 1005: -1
- 1006: 40
- 1007: **1009**  Right Child
- 1008: -1
Summary

• The world of computers
  – 50 years of history
  – ever faster, ever smaller, ever larger memory
  – Moore’s law

• fundamental ideas of von Neumann computer still prevail
  – digital (binary), stored program
  – conceptual organisation

• Flexibility due to (re)programming