Computer architecture

Introduction

http://d3s.mff.cuni.cz/teaching/computer_architecture/

Lubomír Bulej
bulej@d3s.mff.cuni.cz

CHARLES UNIVERSITY IN PRAGUE
faculty of mathematics and physics
What is interesting on computers?

- **Very dynamic field**
  - First electronic computers around 1940
  - 60 years later: pervasive
  - New technologies replaced before they become old

- **Tremendous impact on everyday life**
  - Internet, embedded computers, human genome, computational chemistry, ...
  - New possibilities with every new order of magnitude in cost reduction, performance increase, size reduction
What is a computer?

- **A broad term**
  - Many common technologies
  - Different architecture to match different requirements

- **Main classes**
  - **Personal computers**
    - Optimal price/performance ratio (drives development)
  - **Servers, mainframes, supercomputers**
    - Higher throughput, reliability, computing power
    - Scientific calculations, serving high number of users
  - **Embedded computers**
    - The most rapidly growing market (not only mobile devices)
    - Limited resources (memory, performance, energy, cost), special requirements (sturdiness)
Global Internet Device Installed Base Forecast

Source: Gartner, IDC, Strategy Analytics, Machina Research, company filings, BII estimates
Mainframe (1964)

- **IBM System 360**
  - Integrated circuits
  - Revolutionary elements
    - Modular constructions
    - Unified data and instructions
    - Unified interface for peripheral devices
    - Memory protection
  - Architectural elements kept even in today’s mainframes
Mainframe (2005)

- IBM System Z9-109 model S54
  - 60 configurable LPARS
  - Special-purpose processors
  - 512 GB of memory
  - 1,740 kg, 2.49 m², 18.3 kW input power
  - Availability/reliability, throughput, security
Less common personal computer
Typical personal computer
What’s in the box?

- Motherboard
- Processor
- Memory (RAM, ROM)
- Chipset
- Basic I/O devices
What’s in the box?

- Motherboard
- Processor
- Memory (RAM, ROM)
- Chipset
- Basic I/O devices
- Optical drive
- Hard drive
**What’s in the box?**

- Motherboard
- Processor
- Memory (RAM, ROM)
- Chipset
- Basic I/O devices
- Optical drive
- Hard drive
- Expansion cards
  - Video card
What’s in the box?

- Motherboard
- Processor
- Memory (RAM, ROM)
- Chipset
- Basic I/O devices
- Optical drive
- Hard drive
- Expansion cards
- Video card
- Power supply
Motherboard
Motherboard (2)

North Bridge (memory controller)
- Front-side Bus
- Memory Bus

South Bridge (I/O controller)
- Internal Bus
- Legacy Bus
- SATA
- USB
- Ethernet
- Audio
- CMOS memory

High-Speed Peripherals
- Front-side Bus
- High-Speed Bus

RAM

Peripherals
- Peripheral Bus

Flash ROM

Clock Generator

Super I/O
- Serial
- Parallel
- Floppy
- Keyboard
- Mouse

CPU

Computer Architecture, Introduction, summer 2020/2021
**Processor**

- **Key elements**
  - Data path
    (operates on data)
  - Control
    (controls data path)
  - Memory elements
    (registers and cache)

- **Intel Core i7-980X**
  - 6 cores, 12 MB L3 cache, clock frequency 3.33 GHz
  - 32 nm technology, 248 mm², 1.2 billion transistors

Source: intel.com
Operating memory

- Volatile

  - Running programs and data
  - Directly addressed by the processor
  - *Dynamic Random-Access Memory (DRAM)*
    - Constant access time (tens of nanoseconds)
    - Bits stored as charge in capacitors
      - Needs periodic refresh (16 Hz typical)
    - Capacity in gigabytes

Source: slashgear.com
Operating memory (2)

- **Volatile**
  - *Static Random-Access Memory (SRAM)*
    - Implemented using two-state flip flops (requires 4 to 6 transistors per bit)
      - No need of periodic refresh
      - Significantly faster (units of nanoseconds), significantly lower density, significantly higher cost
    - Processor caches and register
    - Other kinds of processor-internal memory
Processor and memory technology

- **Transistor**
  - Basic building block
    - Discrete (a controllable switch) instead of analog (amplifier) application

- **Integrated circuit**
  - Multiple transistors on a single chip
    - Additional parts (capacitors, resistors, etc.)
  - Better technology → smaller dimensions → higher level of integration → higher processor speed and higher memory capacity
Processor and memory technology (2)
Processor and memory technology (3)

Silicon ingot → Slicer → Blank wafers → 20 to 40 processing steps

Blank wafers → Slicer → Tested dies → Tested wafer → Wafer tester

Wafer tester → Patterned wafers

Bond die to package → Tested dies → Dicer → Tested wafer

Tested wafer → Wafer tester → Patterned wafers

Bond die to package → Tested dies → Dicer

Dicer → Tested wafer

Tested wafer → Wafer tester

Patterned wafers → Wafer tester

Wafer tester → Patterned wafers

Packaged dies → Tested packaged dies

Tested packaged dies → Ship to customers

Source: P&H
Secondary storage

- **Persistent**
  - Data retained without power
  - Data files and executables
  - Not directly addressable by CPU (I/O devices, controlled by a program – operating system)

- **Hard drive**
  - Magnetic rotational medium
  - Sector-based addressing (chunks of 512 B or 4 KB), access times in tens of milliseconds (not constant)

- **Solid-State Drive (SSD), flash memory**
  - Solid (non-moving), transistor-based persistent storage (*floating-gate MOSFET*)
  - Asymmetric read/write operations (read individual bits, write large blocks), constant access time in tens to hundreds of microseconds
Basic computer organization

- **Computer**
  - input
  - output
  - memory
  - processor
    - data path
    - control

- **Technology independent**
  - First both today’s and past computers

Source: P&H
Inputs and outputs

- **Input devices**
  - Keyboard, mouse, tablet, fingerprint reader, joystick, camera, ...

- **Output devices**
  - CRT display, LCD panel, graphic card, printer

- **Input/output devices**
  - Network interface card, hard drive, sound card, camera, force-feedback steering wheel, ...
Graphical screen output

- **Framebuffer (memory on the graphic card)**
  - Every place in memory (or a group of places) corresponds to a pixel on the screen
  - Contents of the place determines color
  - Size of the place determines color resolution

Source: P&H
Below your program
From power-on to running applications

- **Firmware**
  - BIOS (Basic Input/Output System)

- **Operating system loader**
  - Boot sector
  - Boot loader

- **Operating system**

- **User interface/desktop environment**

- **Application**
100s of 1000s of lines of code

- **Application software**
  - Text editor, spread sheet, ...
  - User interface libraries

- **System software**
  - Operating system
    - Input/output operations
    - Memory and storage management
    - Resource sharing
  - Firmware

- **Hardware**
  - Processor, memory, I/O devices
100s of 1000s of lines of code

Source: https://informationisbeautiful.net/visualizations/million-lines-of-code (data as of 2016)
Basic concept of computer architecture

Abstraction
Abstraction

- **Required to bridge semantic gaps**
  - From a concrete (technical) language to an abstract (general) language
  - Expressing the same using more general terms while encapsulating internal details and preserving accuracy
    - More concise and compact expression
  - “An abstraction is one thing that represents several real things equally well.” (Edsger Dijkstra)
From a user to an algorithm

User

Algorithm

Semantic gap

Delete paragraph
Set font

....

document.par[i].value = ...;
document.set_font(...);
...
From an algorithm to a program

Algorithm

document.par[i].value = ...;
document.set_font(...);
...

Semantic gap

Program

MULI $2, $5, 4
ADD $2, $4, $2
LW $16, 0($2)
...

From a program to machine code

Program

MULI $2, $5, 4
ADD $2, $4, $2
LW $16, 0($2)
...

Processor

0101001010010
0110101001101
0111010110101
...

Semantic gap
Example: Swap k-th and (k+1)-th element

High-level programming language

```c
void swap(unsigned int array[], unsigned int k) {
    unsigned int old = array[k];
    array[k] = array[k + 1];
    array[k + 1] = old;
}
```
Example: Swap k-th and (k+1)-th element

Assembler representation for MIPS

swap:
  sll  $a1, $a1, 2
  addu $a1, $a1, $a0
  lw   $v0, 0($a1)
  lw   $v1, 4($a1)
  sw   $v1, 0($a1)
  sw   $v0, 4($a1)
  jr   $ra
Example: Swap k-th and (k+1)-th element

Assembler representation for SuperH

```assembly
swap:
  shll2  r5
  mov   r4,r1
  add   r5,r1
  mov.l @r1,r2
  add   #4,r5
  add   r5,r4
  mov.l @r4,r3
  mov.l r3,@r1
  rts
  mov.l r2,@r4
```
Example: Swap k-th and (k+1)-th element

**Assembler representation for x86-64**

```assembly
swap:
    movslq  %esi, %rsi
    leaq   (%rdi, %rsi, 4), %rdx
    leaq   4(%rdi, %rsi, 4), %rax
    movl   (%rdx), %ecx
    movl   (%rax), %esi
    movl   %esi, (%rdx)
    movl   %esi, (%rdx)
    movl   %ecx, (%rax)
    retq
```
Example: Swap k-th and (k+1)-th element

Machine code for MIPS

00000000000101001010001000000000
000000001010010010000101000000100001
10001100101000100000000000000000000
1000110010100011000000000000000100
1010110010100010000000000000000100
1010110010100011000000000000000000
0000000111110000000000000000001000
Example: Swap k-th and (k+1)-th element

Machine code for SuperH

```
0000100001000101
0100001101100001
0101110000110001
0010010011000010
0000010001110101
0101110000110100
0100001001100011
0011001000100001
0000101100000000
0010001000100100
```

Example: Swap k-th and (k+1)-th element

Machine code for x86-64

010010000110011111111111110110
0100100010001101110001111110110
01001000100011010001101010001101001001110110111
010010001000110101000110101000100001011000100001000
100010110000001010
10001011010110000000
1000100101110010
1000100100001000
11000111
Implementation

- The opposite of abstraction
  - Concretization
  - From computer architecture to concrete computer
  - High-level language
    - Block diagrams, functional description of circuits
  - Low-level language
    - Circuit diagrams connecting electronic components, masks for producing semiconductor elements in an integrated circuit
  - „Machine code“
    - Physical realization of a computer
Abstraction layers in a computer

User interface
Application engine
Application libraries/frameworks
Operating system
Instruction Set Architecture (ISA)
Data path, control
Logic circuits
Transistors

Abstraction level
Software
Applications
Hardware
HW/SW interface
Beware: abstraction is (only) a tool!

### Latency Numbers Every Programmer Should Know

<table>
<thead>
<tr>
<th>Latency (ns)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ns</td>
<td>Main memory reference: 100 ns</td>
</tr>
<tr>
<td>0.5 ns</td>
<td>L1 cache reference</td>
</tr>
<tr>
<td>5 ns</td>
<td>Branch mispredict</td>
</tr>
<tr>
<td>7 ns</td>
<td>L2 cache reference</td>
</tr>
<tr>
<td>25 ns</td>
<td>Mutex lock/unlock</td>
</tr>
<tr>
<td>100 ns</td>
<td></td>
</tr>
</tbody>
</table>

- **Send 1KB over 1Gbps network:** 10 μs
- **SSD random read (10Gb/s SSD):** 150 μs
- **Read 1MB sequentially from memory:** 250 μs
- **Round trip in same datacenter:** 500 μs
- **Packet roundtrip CA to Netherlands:** 150 ns

Source: https://gist.github.com/2841832