CS3350B Computer Architecture
Introduction

Marc Moreno Maza

http://www.csd.uwo.ca/~moreno/cs3350_moreno/index.html
Department of Computer Science
University of Western Ontario, Canada

Thursday January 5, 2017
Konrad Zuse’s Z3 electro-mechanical computer (1941, Germany). Turing complete, though conditional jumps were missing.
Colossus (UK, 1941) was the world’s first totally electronic programmable computing device. But not Turing complete.
Harvard Mark I – IBM ASCC (1944, US). Electro-mechanical computer (no conditional jumps and not Turing complete). It could store 72 numbers, each 23 decimal digits long. It could do three additions or subtractions in a second. A multiplication took six seconds, a division took 15.3 seconds, and a logarithm or a trigonometric function took over one minute. A loop was accomplished by joining the end of the paper tape containing the program back to the beginning of the tape (literally creating a loop).
Electronic Numerical Integrator And Computer (ENIAC). The first general-purpose, electronic computer. It was a Turing-complete, digital computer capable of being reprogrammed and was running at 5,000 cycles per second for operations on the 10-digit numbers.
The IBM Personal Computer, commonly known as the IBM PC (Introduced on August 12, 1981).
The Pentium Family.
<table>
<thead>
<tr>
<th></th>
<th>Size</th>
<th>Line Size</th>
<th>Latency</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L1 Data Cache</strong></td>
<td>32 KB</td>
<td>64 bytes</td>
<td>3 cycles</td>
<td>8-way</td>
</tr>
<tr>
<td><strong>L1 Instruction Cache</strong></td>
<td>32 KB</td>
<td>64 bytes</td>
<td>3 cycles</td>
<td>8-way</td>
</tr>
<tr>
<td><strong>L2 Cache</strong></td>
<td>6 MB</td>
<td>64 bytes</td>
<td>14 cycles</td>
<td>24-way</td>
</tr>
</tbody>
</table>

Capacity
Access Time
Cost

**CPU Registers**
100s Bytes
300 – 500 ps (0.3-0.5 ns)

**L1 and L2 Cache**
10s-100s K Bytes
~1 ns - ~10 ns
$1000s/ GByte

**Main Memory**
G Bytes
80ns- 200ns
~ $100/ GByte

**Disk**
10s T Bytes, 10 ms
(10,000,000 ns)
~ $1 / GByte

**Tape**
infinite
sec-min
~$1 / GByte

**Staging Xfer Unit**
prog./compiler
1-8 bytes

**L1 Cache**
Instr. Operands
Blocks

**L2 Cache**
Blocks

**Memory**
Memory blocks
Blocks

**Disk**
Disk blocks
Blocks

**Tape**
Files

**Upper Level**
faster

**Lower Level**
Large
Once upon a time, every thing was slow in a computer...
Classes of Computers

- **Personal computers**
  - General purpose, variety of software
  - Subject to cost/performance trade-off

- **Server computers**
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized

- **Supercomputers**
  - High-end scientific and engineering calculations
  - Highest capability but represent a small fraction of the overall computer market

- **Embedded computers**
  - Hidden as components of systems
  - Stringent power/performance/cost constraints
Components of a computer

- Same components for all kinds of computer
  - desktop, server, embedded
Below your program

- **Application software**
  - Written in a high-level language

- **System software**
  - Compiler: translates HLL code to machine code
  - Operating system: service code
  - Handling input/output
  - Managing memory and storage
  - Scheduling tasks & sharing resources

- **Hardware**
  - Processor, memory, I/O controllers
Levels of program code

- **High-level language**
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability

- **Assembly language**
  - Textual representation of instructions

- **Hardware representation**
  - Binary digits (bits)
  - Encoded instructions and data
Old-school machine structures (layers of abstraction)
New-school machine structures

Software

- Parallel Requests
  Assigned to computer
e.g., Search “Katz”

- Parallel Threads
  Assigned to core
e.g., Look-up, Ads

- Parallel Instructions
  >1 instruction @ one time
e.g., 5 pipelined instructions

- Parallel Data
  >1 data item @ one time
e.g., Add of 4 pairs of words

- Hardware descriptions
  All gates working in parallel at same time

Hardware

Harness Parallelism & Achieve High Performance

Warehouse
Scale
Computer

Smart Phone

Computer

Core

Memory

(Cache)

Input/Output

Instruction Unit(s)

Core

Functional Unit(s)

Main Memory

Logic Gates
Why do computers become so complicated?

Pursuing performance!

- Eight great ideas
  - Use abstraction to simplify design
  - Design for Moore’s Law
  - Make the common case fast
  - Performance via parallelism
  - Performance via pipelining
  - Performance via prediction
  - Hierarchy of memories
  - Dependability via redundancy
Great Idea #1: Abstraction

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;

lw $t0, 0($2)
lw $t1, 4($2)
sw $t1, 0($2)
sw $t0, 4($2)
```

# Anything can be represented as a number, i.e., data or instructions

```
0000 1001 1100 0110 1010 1111
0101 1000
1010 1111 0101 1000 0000 1001
1100 0110
1100 0110 1010 1111 0101 1000
```
Great idea #2: Moore’s Law

Predicts: 2X Transistors / chip every 2 years

Gordon Moore
Intel Cofounder
B.S. Cal 1950!
Great idea #4: Performance via parallelism
Great idea #5: Performance via pipelining

In time slot 3, instruction 1 is being executed, instruction 2 is in the operand fetch phase, and instruction 3 is being fetched from memory.
Caveat: Amdahl’s Law

**Fig 3** Amdahl’s Law an Obstacle to Improved Performance  Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl’s Law is a major obstacle in boosting multicore microprocessor performance. Diagram assumes no overhead in parallel processing. Years shown for design rules based on Intel planned and actual technology. Core count assumed to double for each rule generation.
Great idea #7: Memory hierarchy (principle of locality)
Great Idea #8: Dependability via redundancy

- Redundancy so that a failing piece doesn’t make the whole system fail

- Increasing transistor density reduces the cost of redundancy
Great Idea #8: Dependability via redundancy

Applies to everything from data centers to storage to memory to instructors

- Redundant data centers so that can lose 1 datacenter but Internet service stays online
- Redundant disks so that can lose 1 disk but not lose data (Redundant Arrays of Independent Disks/RAID)
- Redundant memory bits of so that can lose 1 bit but no data (Error Correcting Code/ECC Memory)
Understanding performance

- **Algorithm**
  Determines number of operations executed

- **Programming language, compiler, architecture**
  Determine number of machine instructions executed per operation

- **Processor and memory system**
  Determine how fast instructions are executed

- **I/O system (including OS)**
  Determines how fast I/O operations are executed
What you will learn

- How programs are translated into the machine language, and how the hardware executes them
- The hardware/software interface
- What determines program performance, and how it can be improved
- How hardware designers improve performance
- What is parallel processing
Course Topics

1. Introduction
   - Machine structures: layers of abstraction
   - Eight great ideas

2. Performance Metrics I
   - CPU performance
   - perf, a profiling tool

3. Memory Hierarchy
   - The principle of locality
   - DRAM and cache
   - Cache misses
   - Performance metrics II: memory performance and profiling
   - Cache design and cache mapping techniques

4. MIPS Instruction Set Architecture (ISA)
   - MIPS number representation
   - MIPS instruction format, addressing modes and procedures
   - SPIM assembler and simulator
Course Topics (cont’d)

5. Introduction to Logic Circuit Design
   - Switches and transistors
   - State circuits
   - Combinational logic circuits
   - Combinational logic blocks
   - MIPS single cycle and multiple cycle CPU data-path and control

6. Instruction Level Parallelism
   - Pipelining the MIPS ISA
   - Pipelining hazards and solutions
   - Multiple issue processors
   - Loop unrolling, SSE

7. Multicore Architecture
   - Multicore organization
   - Memory consistency and cache coherence
   - Thread level parallelism

8. GPU Architecture
   - Memory model
   - Execution model: scheduling and synchronization
Student evaluation

- Four assignments, each worth 10% of the final mark
  - Assignment 1 (CPU performance and memory hierarchy), due Friday, Jan. 27
  - Assignment 2 (MIPS and logic circuits), due Friday, Feb. 17
  - Assignment 3 (Circuits and data-path), due Friday, March 10,
  - Assignment 4 (ILP and multicore), due Friday, March 31.

- Four quizzes (key concepts, 30-minute in class), each worth 5% of the final mark
  - Quiz 1 (CPU/memory performance metrics and hierarchical memory), Thursday, Jan. 26
  - Quiz 2 (MIPS), Thursday, Feb. 16
  - Quiz 3 (Circuits and data-path), Thursday, March 9
  - Quiz 4 (ILP and multicore), Thursday, March 30

- One final exam (covering all the course contents), worth 40% of the final mark
Recommended (but not required) textbook


Instructor: Marc Moreno Maza

- Email: moreno@csd.uwo.ca
- Office room: MC327
- Office hours:
  - Tuesdays 2:30pm - 3:15pm
  - Thursdays 1:30pm - 3:15pm
  - Otherwise by appointment

Teaching Assistants: Davood Mohajerani and Egor Chesakov

- Emails: dmohajer@uwo.ca and echesako@uwo.ca
- Office room: MC327
- Office hours:
  - Mondays 1:30pm - 3:30pm
  - Wednesdays 1:30pm - 3:30pm
Acknowledgements

The lecturing slides of this course are adapted from the slides accompanied with the text book and the teaching materials posted on the www by other instructors who are teaching Computer Architecture courses.