Computer Systems Overview

ECE 154B
Dmitri Strukov
Outline

• Course information
• Trends
• Computing classes
• Quantitative Principles of Design
• Dependability
Course organization

• Class website: http://www.ece.ucsb.edu/~strukov/ece154b Winter2014/home.htm

• Instructor office hours: Wed, 2:00 pm – 4:00 pm

• Michael Klachko (TA) office hours: by appointment; michaelklachko@gmail.com
Textbook


Class topics

• Computer fundamentals (historical trends, performance) – 1 week
• Memory hierarchy design - 2 weeks
• Instruction level parallelism (static and dynamic scheduling, speculation) – 2 weeks
• Data level parallelism (vector, SIMD and GPUs) – 2 weeks
• Thread level parallelism (shared-memory architectures, synchronization and cache coherence) – 2 weeks
• Warehouse-scale computers or Detailed analysis of some specific uP (1 week)
Grading

• Projects: 50 %
• Midterm: 20 %
• Final: 30 %

• Project course work will involve program performance analysis and architectural optimizations for superscalar processors using SimpleScalar simulation tools

• HW will be assigned each week but not graded
Course prerequisites

- ECE 154A or equivalent
ENIAC: Electronic Numerical Integrator And Computer, 1946
VLSI Developments

1946: ENIAC electronic numerical integrator and computer

• Floor area
  – 140 m²

• Performance
  – multiplication of two 10-digit numbers in 2 ms

2011: High Performance microprocessor

• Chip area
  – 100-400 mm² (for multi-core)

• Board area
  – 200 cm²; improvement of $10^4$

• Performance:
  – 64 bit multiply in few ns; improvement of $10^6$
Computer trends: Performance of a (single) processor

Move to multi-processor

RISC

25%/year

1.5, VAX-11/785

22%/year

52%/year

Intel Xeon 6 cores, 3.3 GHz (boost to 3.6 GHz)
Intel Xeon 4 cores, 3.3 GHz (boost to 3.6 GHz)
Intel Core i7 Extreme 4 cores 2.8 GHz (boost to 3.5 GHz)
Intel Core Duo Extreme 2 cores, 3.0 GHz
Intel Core 2 Extreme 2 cores, 2.9 GHz
AMD Athlon 64 2.8 GHz
AMD Athlon, 2.5 GHz
AMD Athlon, 2.3 GHz

Intel Xeon EE 3.2 GHz

Intel Xeon 3.0 GHz

IBM Power4, 1.3 GHz

AlphaServer 9000 600 MHz
Digital AlphaStation 5/600, 600 MHz
Digital AlphaStation 4/266, 266 MHz
Digital AlphaStation 3/120, 120 MHz
Digital AlphaStation 2/60, 60 MHz
HP 9000/750, 75 MHz
IBM RISCstation 100, 150 MHz
MIPS R2000, 25 MHz
MIPS R12000, 25 MHz

Sun-4/280, 18.7 MHz

VAX 8700, 22 MHz

AX-11/780, 5 MHz

Intel D850EMVR motherboard (3.06 GHz, Pentium 4 processor with Hyper-Threading Technology)
Digital AlphaServer 8400 1/575, 575 MHz
IBM POWERstation 100, 150 MHz
Digital 3000 AXP/500, 150 MHz
Digital 3000 AXP/500, 500 MHz
Digital AlphaStation 5/500, 500 MHz
Digital AlphaStation 5/300, 300 MHz
IBM POWERstation 100, 150 MHz
Digital AlphaStation 4/266, 266 MHz
Digital AlphaStation 3/120, 120 MHz
Digital AlphaStation 2/60, 60 MHz
HP 9000/750, 75 MHz
IBM RISCstation 100, 150 MHz
MIPS R2000, 25 MHz
MIPS R12000, 25 MHz

Sun-4/280, 18.7 MHz

VAX 8700, 22 MHz

AX-11/780, 5 MHz

Intel D850EMVR motherboard (3.06 GHz, Pentium 4 processor with Hyper-Threading Technology)
Digital AlphaServer 8400 1/575, 575 MHz
IBM POWERstation 100, 150 MHz
Digital 3000 AXP/500, 150 MHz
Digital 3000 AXP/500, 500 MHz
Digital AlphaStation 5/500, 500 MHz
Digital AlphaStation 5/300, 300 MHz
IBM RISCstation 100, 150 MHz
MIPS R2000, 25 MHz
MIPS R12000, 25 MHz

Sun-4/280, 18.7 MHz

VAX 8700, 22 MHz

AX-11/780, 5 MHz
Current Trends in Architecture

• Cannot continue to leverage Instruction-Level parallelism (ILP)
  – Single processor performance improvement ended in 2003

• New models for performance:
  – Data-level parallelism (DLP)
  – Thread-level parallelism (TLP)
  – Request-level parallelism (RLP)

• These require explicit restructuring of the application
Classes of Computers

• Personal Mobile Device (PMD)
  – e.g. start phones, tablet computers
  – Emphasis on energy efficiency and real-time

• Desktop Computing
  – Emphasis on price-performance

• Servers
  – Emphasis on availability, scalability, throughput

• Clusters / Warehouse Scale Computers
  – Used for “Software as a Service (SaaS)”
  – Emphasis on availability and price-performance
  – Sub-class: Supercomputers, emphasis: floating-point performance and fast internal networks

• Embedded Computers
  – Emphasis: price
Defining Computer Architecture

• “Old” view of computer architecture:
  – Instruction Set Architecture (ISA) design
  – i.e. decisions regarding:
    • registers, memory addressing, addressing modes, instruction operands, available operations, control flow instructions, instruction encoding

• “Real” computer architecture:
  – Specific requirements of the target machine
  – Design to maximize performance within constraints: cost, power, and availability
  – Includes ISA, microarchitecture, hardware
Trends in Technology

• Integrated circuit technology
  – Transistor density: 35%/year
  – Die size: 10-20%/year
  – Integration overall: 40-55%/year

• DRAM capacity: 25-40%/year (slowing)

• Flash capacity: 50-60%/year
  – 15-20X cheaper/bit than DRAM

• Magnetic disk technology: 40%/year
  – 15-25X cheaper/bit than Flash
  – 300-500X cheaper/bit than DRAM
CMOS improvements:
- Transistor density: $4x / 3$ yrs
- Die size: $10-25\% / \text{yr}$
Bandwidth and Latency

• Bandwidth or throughput
  – Total work done in a given time
  – 10,000-25,000X improvement for processors
  – 300-1200X improvement for memory and disks

• Latency or response time
  – Time between start and completion of an event
  – 30-80X improvement for processors
  – 6-8X improvement for memory and disks
Bandwidth and Latency

CPU high, Memory low ("Memory Wall")

Log-log plot of bandwidth and latency milestones

Performance Milestones

- Ethernet: 10Mb, 100Mb, 1000Mb, 10000 Mb/s (16x, 1000x)
- Memory Module: 16bit plain DRAM, Page Mode DRAM, 32b, 64b, SDRAM, DDR SDRAM (4x, 120x)
- Disk: 3600, 5400, 7200, 10000, 15000 RPM (8x, 143x)
Transistors and Wires

- Feature size
  - Minimum size of transistor or wire in x or y dimension
  - 10 microns in 1971 to .032 microns in 2011
  - Transistor performance scales linearly
    - Wire delay does not improve with feature size!
  - Integration density scales quadratically
Scaling with Feature Size

- If $s$ is scaling factor, then density scale as $s^2$
- Logic gate capacitance $C$ (traditionally dominating): $\sim 1/s$
- Capacitance of wires
  - fixed length: $\sim$does not change
  - reduced length by $s$: $\sim 1/s$
- Resistance of wires
  - fixed length: $s^2$
  - reduced length by $s$: $s$
- Saturation current $I_{\text{ON}}$ (which is reciprocal of effective $R_{\text{ON}}$ of the gate): $1/s$
- Voltage $V$: $\sim 1/s$
  (does not scale as fast anymore because of subthreshold leakage)
- Gate delay: $\sim CV/I_{\text{ON}} = 1/s$
Power and Energy

• Problem: Get power in, get power out

• Thermal Design Power (TDP)
  – Characterizes sustained power consumption
  – Used as target for power supply and cooling system
  – Lower than peak power, higher than average power consumption

• Clock rate can be reduced dynamically to limit power consumption

• Energy per task is often a better measurement
Dynamic Energy and Power

• Dynamic energy
  – Transistor switch from 0 -> 1 or 1 -> 0
  – $\frac{1}{2} \times \text{Capacitive load} \times \text{Voltage}^2$

• Dynamic power
  – $\frac{1}{2} \times \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency} \text{ switched}$

• Reducing clock rate reduces power, not energy
Power

- Intel 80386 consumed ~ 2 W
- 3.3 GHz Intel Core i7 consumes 130 W
- Heat must be dissipated from 1.5 x 1.5 cm chip
- This is the limit of what can be cooled by air
Power consumption

Exponential growth
Power doubling per 36 months

Year

CPU Thermal Design Power (Watts)

Intel 486
Pentium
Pentium II & III
Pentium 4, D & M
Core 2 Duo & Quad

Power consumption
Reducing Power

• Techniques for reducing power:
  – Do nothing well
  – Dynamic Voltage-Frequency Scaling
  – Low power state for DRAM, disks
  – Overclocking, turning off cores

Since $I_{\text{ON}} \sim V^2$ and gate delay is $\sim CV/I_{\text{ON}}$, in the first approximation clock frequency (which is reciprocal of gate delay) is proportional to $V$

Lowering voltage reduces the dynamic power consumption and energy per operation but decrease performance because of negative effect on frequency
Static Power

• Static power consumption
  – $\text{Current}_{\text{static}} \times \text{Voltage}$
  – Scales with number of transistors
  – To reduce: power gating
Trends in Cost

• Cost driven down by learning curve
  – Yield

• DRAM: price closely tracks cost

• Microprocessors: price depends on volume
  – 10% less for each doubling of volume
8” MIPS64 R20K wafer (564 dies)

Drawing single-crystal Si ingot from furnace....

Then, slice into wafers and pattern it....
What's the price of an IC?

\[
\text{IC cost} = \frac{\text{Die cost} + \text{Testing cost} + \text{Packaging cost}}{\text{Final test yield}}
\]

**Final test yield:** fraction of packaged dies which pass the final testing state
Integrated Circuits Costs

IC cost = \frac{\text{Die cost} + \text{Testing cost} + \text{Packaging cost}}{\text{Final test yield}}

\text{Die cost} = \frac{\text{Wafer cost}}{\text{Dies per Wafer} \times \text{Die yield}}

\text{Final test yield: fraction of packaged dies which pass the final testing state}

\text{Die yield: fraction of good dies on a wafer}
What's the price of the final product?

- **Component Costs**
- **Direct Costs** (add 25% to 40%) recurring costs: labor, purchasing, warranty
- **Gross Margin** (add 82% to 186%) nonrecurring costs: R&D, marketing, sales, equipment maintenance, rental, financing cost, pretax profits, taxes
- **Average Discount** to get List Price (add 33% to 66%): volume discounts and/or retailer markup
Integrated Circuit Cost

• Integrated circuit

\[
\text{Cost of integrated circuit} = \frac{\text{Cost of die} + \text{Cost of testing die} + \text{Cost of packaging and final test}}{\text{Final test yield}}
\]

\[
\text{Cost of die} = \frac{\text{Cost of wafer}}{\text{Dies per wafer} \times \text{Die yield}}
\]

\[
\text{Dies per wafer} = \frac{\pi \times (\text{Wafer diameter}/2)^2}{\text{Die area}} - \frac{\pi \times \text{Wafer diameter}}{\sqrt{2} \times \text{Die area}}
\]

• Bose-Einstein formula:

\[
\text{Die yield} = \text{Wafer yield} \times \frac{1}{1 + \text{Defects per unit area} \times \text{Die area}^N}
\]

• Defects per unit area = 0.016-0.057 defects per square cm (2010)
• \(N = \text{process-complexity factor} = 11.5-15.5\) (40 nm, 2010)
Quantitative Principles of Design

• Take Advantage of Parallelism

• Principle of Locality

• Focus on the Common Case
  – Amdahl’s Law
  – E.g. common case supported by special hardware; uncommon cases in software

• The Performance Equation
Measuring Performance

• Typical performance metrics:
  – Response time
  – Throughput

• Speedup of X relative to Y
  – Execution time_Y / Execution time_X

• Execution time
  – Wall clock time: includes all system overheads
  – CPU time: only computation time

• Benchmarks
  – Kernels (e.g. matrix multiply)
  – Toy programs (e.g. sorting)
  – Synthetic benchmarks (e.g. Dhrystone)
  – Benchmark suites (e.g. SPEC06fp, TPC-C)
1. Parallelism

How to improve performance?

• (Super)-pipelining
• Powerful instructions
  – MD-technique
    • multiple data operands per operation
  – MO-technique
    • multiple operations per instruction
• Multiple instruction issue
  – single instruction-program stream
  – multiple streams (or programs, or tasks)
Flynn’s Taxonomy

• Single instruction stream, single data stream (SISD)

• Single instruction stream, multiple data streams (SIMD)
  – Vector architectures
  – Multimedia extensions
  – Graphics processor units

• Multiple instruction streams, single data stream (MISD)
  – No commercial implementation

• Multiple instruction streams, multiple data streams (MIMD)
  – Tightly-coupled MIMD
  – Loosely-coupled MIMD
MIPS Pipeline

Five stages, one step per stage
1. IF: Instruction fetch from memory
2. ID: Instruction decode & register read
3. EX: Execute operation or calculate address
4. MEM: Access memory operand
5. WB: Write result back to register
Review from Last Lecture

![Task-time diagram](a) Task-time diagram

![Space-time diagram](b) Space-time diagram

**Execution time = 1/ Performance = Inst count x CPI x CCT**

\[ N = \text{# of stages for pipeline design or } \sim \text{ maximum number of steps for MC} \]

\[
P_{\text{ideal MCP}} = \frac{N}{\text{InstCount}} + 1 - \frac{1}{\text{InstCount}}
\]

→ large N and/or small InstCount result in worse CPI

→ Performance to run one instruction is the same as of CP (i.e. latency for single instruction is not reduced)

<table>
<thead>
<tr>
<th>Design</th>
<th>Inst count</th>
<th>CPI</th>
<th>CCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Cycle (SC)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Multi cycle (MC)</td>
<td>1</td>
<td>( N \geq \text{CPI} &gt; 1 ) (closer to N than 1)</td>
<td>( &gt; 1/N )</td>
</tr>
<tr>
<td>Multi cycle pipelined (MCP)</td>
<td>1</td>
<td>&gt; 1</td>
<td>&gt;1/N</td>
</tr>
</tbody>
</table>

What are the other issues affecting CCT and CPI for MC and MCP?
Pipelined Instruction Execution

Time (clock cycles)

Cycle 1
- Ifetch
- Reg
- ALU

Cycle 2
- Ifetch
- Reg
- ALU

Cycle 3
- Ifetch
- Reg
- ALU

Cycle 4
- DMem
- Reg

Cycle 5
- DMem
- Reg

Cycle 6
- DMem
- Reg

Cycle 7
- DMem
- Reg

Instr. Order

Instr. Order
Limits to pipelining

- **Hazard**s prevent next instruction from executing during its designated clock cycle
  - **Structural hazards**: attempt to use the same hardware to do two different things at once
  - **Data hazards**: Instruction depends on result of prior instruction still in the pipeline
  - **Control hazards**: Caused by delay between the fetching of instructions and decisions about changes in control flow (branches and jumps).
2. The Principle of Locality

- Programs access a relatively small portion of the address space at any instant of time.
- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon (e.g., straight-line code, array access)
- Last 30 years, HW relied on locality for memory perf.
Memory Hierarchy Levels

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

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

- **Main Memory**
  - G Bytes
  - 80ns - 200ns
  - ~ $10/ GByte

- **Disk**
  - 10s T Bytes, 10 ms
  - (10,000,000 ns)
  - ~ $0.1 / GByte

- **Tape**
  - infinite sec-min
  - ~$0.1 / GByte

---

**Staging Xfer Unit**

- **Upper Level**
  - faster
  - prog./compiler
    - 1-8 bytes
  - cache cntl
    - 32-64 bytes
  - cache cntl
    - 64-128 bytes
  - OS
    - 4K-8K bytes

- **Lower Level**
  - Larger
  - user/operator
    - Gbytes
  - still needed?
3. Focus on the Common Case

• Favor the frequent case over the infrequent case
  – E.g., Instruction fetch and decode unit used more frequently than multiplier, so optimize it first
  – E.g., If database server has 50 disks / processor, storage dependability dominates system dependability, so optimize it first

• Frequent case is often simpler and can be done faster than the infrequent case
  – E.g., overflow is rare when adding 2 numbers, so improve performance by optimizing more common case of no overflow
  – May slow down overflow, but overall performance improved by optimizing for the normal case

• What is frequent case? How much performance improved by making case faster? => Amdahl’s Law
Amdahl’s Law

\[ \text{Speedup}_{\text{overall}} = \frac{T_{\text{exec,old}}}{T_{\text{exec,new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}} \]
Amdahl’s Law

• Floating point instructions improved to run 2 times faster, but only 10% of actual instructions are FP

\[ T_{\text{exec,new}} = \]

\[ \text{Speedup}_{\text{overall}} = \]
Amdahl’s Law

• Floating point instructions improved to run 2X; but only 10% of actual instructions are FP

\[
T_{exec,new} = T_{exec,old} \times (0.9 + 0.1/2) = 0.95 \times T_{exec,old}
\]

\[
\text{Speedup}_{overall} = \frac{1}{0.95} = 1.053
\]
Amdahl's law
Principles of Computer Design

• The Processor Performance Equation

\[ \text{CPU time} = \text{CPU clock cycles for a program} \times \text{Clock cycle time} \]

\[ \text{CPU time} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}} \]

\[ \text{CPI} = \frac{\text{CPU clock cycles for a program}}{\text{Instruction count}} \]

\[ \text{CPU time} = \text{Instruction count} \times \text{Cycles per instruction} \times \text{Clock cycle time} \]

\[ \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}} = \frac{\text{Seconds}}{\text{Program}} = \text{CPU time} \]
Principles of Computer Design

- Different instruction types having different CPIs

\[
\text{CPU clock cycles} = \sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i
\]

\[
\text{CPU time} = \left( \sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i \right) \times \text{Clock cycle time}
\]
Dependability

• Module reliability
  – Mean time to failure (MTTF)
  – Mean time to repair (MTTR)
  – Mean time between failures (MTBF) = MTTF + MTTR
  – Availability = MTTF / MTBF
Acknowledgements

Some of the slides contain material developed and copyrighted by Henk Corporaal (TU/e) and instructor material for the textbook.