CMSC 411
Computer Systems Architecture
Lecture 2
Computer Design and Evaluation

Alan Sussman
als@cs.umd.edu

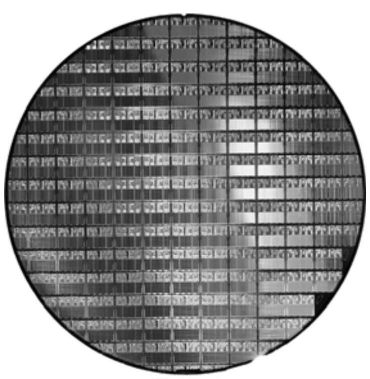
Administrivia

- **Unit 1 Homework due Thursday, 2/12**
  - posted on homework/project web page later today
  - as stated on syllabus, 2 problems will be graded
- **Finish reading Ch. 1 of H&P**

Manufacture of DRAM and other chips

- **Chips are manufactured on wafers - circular disks containing many dies (chips).**
- **The wafer is tested and chopped into dies.**

Fig. 1.12 in H&P
117 AMD Opterons



Wafers and dies

- **To find the cost of a die:**
  - **Number of dies per wafer is *at most* the area of the wafer divided by the area of the die.**
  - **The cost of the wafer divided by the number of working dies per wafer is the cost of each die.**
- **The fraction of working dies is called the *die yield*, which decreases as the area of the die increases.**
- **Rule of thumb (p. 20): Cost of die is proportional to the square of the die area**

Comparing performance of two machines

- Definition: Performance is equal to 1 divided by execution time
- Problem: How to measure execution time?

What is time?

- Unix time command example:
  - 90.7u 12.9s 2:39 65%
  - The user used the CPU for 90.7 seconds (user CPU time)
  - The system used it for 12.9 seconds (system CPU time)
  - Elapsed time from the user's request to completion of the task was 2 minutes, 39 seconds (159 seconds)
  - And (90.7 + 12.9)/159 = 65%
    - » the rest of the time was spent waiting for I/O or running other programs

Time (cont.)

- Usual measurements of time:
  - system performance measures the elapsed time on unloaded (single user) system
  - CPU performance measures user CPU time on unloaded system

How to measure CPU performance

- Benchmark: a program used to measure performance
  - real programs - what is reality?
  - kernels - loops in which most of time is spent in a real program
  - toy programs
  - synthetic programs
- Fact: Computer manufacturers tune their product to the popular benchmarks
  - ``Your results may vary," unless you run benchmark programs and nothing else
  - See Figure 1.13 listing programs in the SPEC CPU2006 benchmark suite

Reproducibility

- **Benchmarking is a laboratory experiment, and needs to be documented as fully as a well-run chemistry experiment**
  - **Identify each variable hardware component**
  - **Identify compiler flags and measure variability**
  - **Verify reproducibility and provide data for others to reproduce the benchmarking results**

Reporting results

- **Example of performance for SPEC CFP2000 benchmark is in H&P Figure 1.14**
  - **for Sun Ultra5, AMD Opteron, Intel Itanium 2**
  - **need to look on SPEC web site for all the parameters of the machines, including**
    - » **data on hardware - CPU (how many, primary and secondary caches), memory, disk**
    - » **data on software – OS, compilers, file system type, and compiler flags used (very important!)**
- **How to summarize results?**

Sample results

From Figure 1.15 in H&P 3/e – which machine is fastest?

	Computer A	Computer B	Computer C
Program P1 (sec)	1	10	20
Program P2 (sec)	1000	100	20
Program P3 (sec)	1001	110	40

Statistical reporting

- **"It is rare indeed when advertisers, politicians, pop economists, and drumbeaters for medical programs offer a statistical argument that is not either misleading or downright deceptive." - Martin Gardner**
- **"... in mathematics you don't understand things, you just get used to them." - John von Neumann**
- **"... if you torture your data long enough, they will tell you what you want to hear." - James L. Mills, M.D.**

Statistical reporting (cont.)

- The average execution time is the arithmetic mean:
  - sum the execution times for each program and divide by the number of programs *n*
- The harmonic mean is *n* divided by the sum of some function of each execution time
- Often, both of these means are modified to give more weight to more important programs

Statistical reporting (cont.)

- Arithmetic mean is good if the weights represent your own workload - then you can compare how each machine will perform for you
  - *Don't* set the weights based on performance on any particular machine; can get confusing results by doing that
- Harmonic mean gives same sort of information, but is used when rates (e.g., instructions per second) are measured instead of times

Geometric mean

- To measure relative performance, use the geometric mean
  - Choose a *reference machine*, divide all execution times by the corresponding times on the reference machine, multiply those ratios together, and take the *n*th root of the product
- Geometric means have the nice property that you get consistent results (relative performance) regardless of which machine is used to normalize (Figure 1.14)

How to make computers faster

- Make the common case faster!
- Example: Put more effort and funds into optimizing the hardware for addition than to optimize square root
- Amdahl's law quantifies this principle:
  - Define speedup as the time the task took originally divided by the time the task takes after improvement

Amdahl's Law

- Then Amdahl tells us what the speedup of a particular task is, given
  - fraction *f* of the original execution time that the task could use the improvement
  - the speedup *s* of a task that always uses the improvement
- Then what is the speedup of the task?

Amdahl's Law (cont.)

- Suppose that the original task runs for 1 second so it takes *f* seconds in the critical piece, and *1-f* in other things
- Then the task on the improved machine will take only *f/s* seconds in the critical piece, but will still take *1-f* seconds in other things
- speedup = $\frac{old\_time}{new\_time} = \frac{1}{(1-f)+\frac{f}{s}}$

Example 1

- Suppose we work very hard improving the square root hardware, and that our task originally spends *1%* of its time doing square roots. Even if the improvement reduces the square root time to zero, the speedup is no better than
- speedup = $\frac{1}{1-f} = \frac{1}{.99} = 1.01$
- ***And we might be better off putting our effort into a more important part of the task***

Example 2

- Suppose that, for the same cost, we can speed up integer arithmetic by a factor of 20, or speed up floating point arithmetic by a factor of 2. If our task spends *10%* of its time in integer arithmetic, and *40%* of its time in floating point arithmetic, which should we do?

Example 2 (cont.)

- **Option 1:**
  speedup = $\frac{1}{.9+\frac{.1}{20}}$ **= 1.105**
- **Option 2:**
  speedup = $\frac{1}{.6+\frac{.4}{2}}$ **= 1.25**

CPU Performance

- **More jargon …**
- **Something new happens in a computer during every clock cycle**
- **The clock rate is what manufacturers usually advertise to indicate a chip's speed:**
  - **e.g., a 2.8GHz Pentium 4**
- **But how fast the machine is depends on the clock rate *and* the number of clock cycles per instruction (CPI)**

CPU Performance (cont.)

- **So total CPU time = instruction count (IC) times CPI divided by clock rate (MHz/GHz)**
  - **IC * CPI / GHz**
- **There's an example on p. 43 that illustrates the overall effect**
- **Note: CPI is not a good measure of performance all by itself since it varies by type of instruction!**

How to design a fast computer

- **Amdahl's law says put your effort into optimizing instructions that are often used.**
- **The locality of reference rule-of-thumb says that a program spends *90%* of its time in *10%* of the code, so we should put effort into taking advantage of this, through a memory hierarchy**
- **For data accesses, 2 types of locality**
  - **temporal**
  - **spatial**

Take advantage of parallelism

- At system level
  - e.g., use multiple CPUs (Unit 7), disks (Unit 6)
- At processor level
  - among instructions (Unit 4)
  - pipelining (Unit 3)
- At digital design level
  - e.g., set associative caches (Unit 5), carry-lookahead in ALU design

Performance/Price Performance

For desktop systems (H&P Figure 1.15) – 1GB ECC SDRAM, August 2005

Model	Processor	Clock rate (GHz)	Price
Dell 380	Intel PIV Xeon	3.8	$3346
HP BL25p	AMD Opteron 252	2.6	$3099
HP ML350 G4	Intel PIV Xeon	3.4	$2907
HP rx2620-2	Intel Itanium 2	1.6	$5201
Sun Java WS W1100z	AMD Opteron 150	2.4	$2145

SPEC CINT2000 & CFP2000 – H&P 1.16