COMPUTER ORGANIZATION AND DESIGN
The Hardware/Software Interface

EECS2021

Computer Organization
Fall 2015

The slides are based on the publisher slides and contribution from Profs Amir Asif and Peter Lian
The slides will be modified, annotated, explained on the board, and sometimes corrected in the class

EECS2021 Computer Organization

- Computer Organization and Design– The hardware/Software approach – ARM Edition
- Patterson and John Hennessy
- Morgan kaufmann
- Assessment:
  - Participation 5%
  - Quizzes 20%
  - Lab 20%
  - Midterm 20%
  - Final 35%

EECS2021 Computer Organization

- Monday and Wednesday 5:30-7:00pm
- LAS B
- 2 Lab sections
  - Monday, Tuesday 7-10pm LAS 1006
- Labs are posted on the course web page

EECS2021

- Instructor
  - Mokhtar Aboelaze
  - Office LAS2026 Phone ext: 40607
- Research interests
  - Computer Architecture
  - Low power architecture
  - Embedded systems
  - FPGA (in embedded applications)

Topics

- Computer abstraction and technology – Ch 1
- Instruction language of the computer – Ch 2
- Verilog -- Notes
- Arithmetic for computers – Ch 3
- The processor – Ch 4
- Memory hierarchy and I/O – Ch 5

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 the ALU works and how to improve its performance using pipelining.
- Memory Hierarchy and I/O (basic operation)

Why EECS2021

- Required
- You can not call yourself ECE or CS if you don't know pipelining, assembly, how to measure performance, or how to report performance.
- Knowledge of hardware helps you write better programs

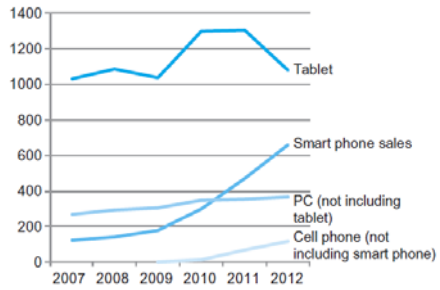
The Computer Revolution

- Progress in computer technology
  - Underpinned by Moore's Law
- Makes novel applications feasible
  - Computers in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search Engines
- Computers are pervasive

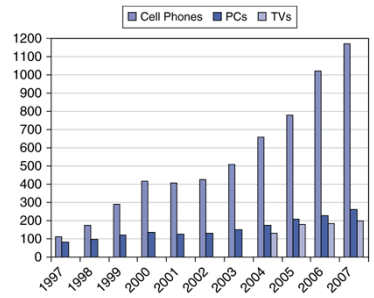
Classes of Computers

- Desktop computers
  - General purpose, variety of software
  - Subject to cost/performance tradeoff
- Server computers
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized
- Embedded computers
  - Hidden as components of systems
  - Stringent power/performance/cost constraints

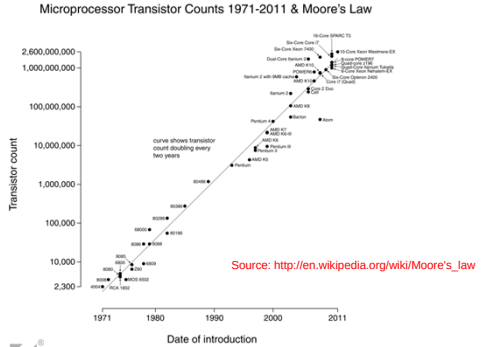
The Processor Market



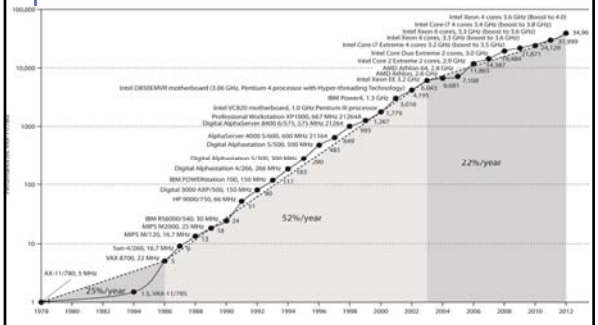
The Processor Market



Trends -- Moore's law

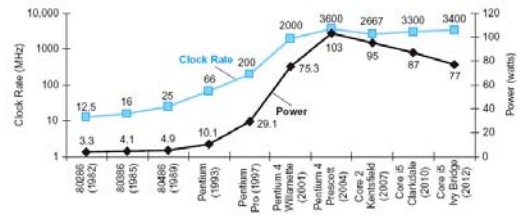


Trends -- Performance



Power

- *Power = Switching Activity × Capacitance × V² × f*
- Static power and its effect

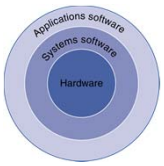


Eight Great Ideas

1. Design for Moore's Law
2. Use Abstraction to Simplify Design
3. Make the Common case fast
4. Performance via Parallelism
5. Performance via Pipelining
6. Performance via Prediction
7. Hierarchy of Memories
8. Dependability via Redundancy

Below Your Program

- Application software
  - Written in 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

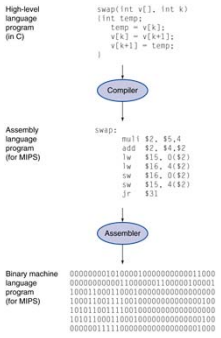


How computers work?

- In your first year, you studies programming (java)
- Sequence of instructions
- Translated to machine language
- Instructions are fetched from the memory one after the other and executed

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



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

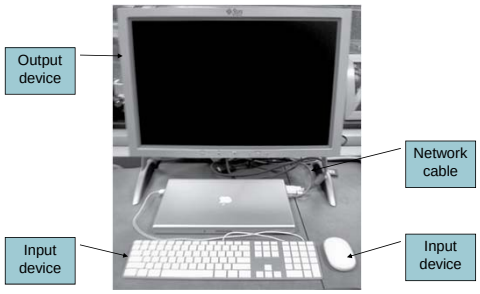
Components of a Computer

§1.3 Under the Covers

The BIG Picture

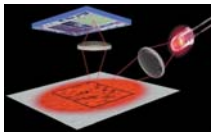
- Same components for all kinds of computer
  - Desktop, server, embedded
- Input/output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers

Anatomy of a Computer



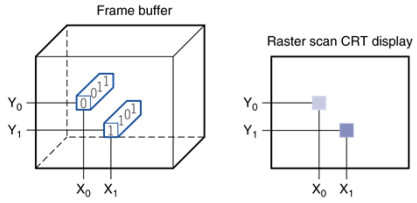
Anatomy of a Mouse

- Optical mouse
  - LED illuminates desktop
  - Small low-res camera
  - Basic image processor
    - Looks for x, y movement
  - Buttons & wheel
- Supersedes roller-ball mechanical mouse

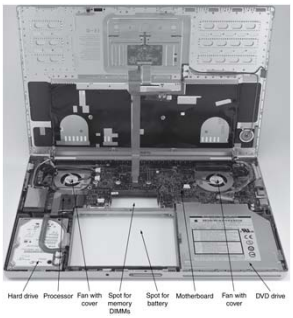


Through the Looking Glass

- LCD screen: picture elements (pixels)
  - Mirrors content of frame buffer memory

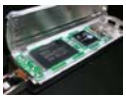


Opening the Box



A Safe Place for Data

- Volatile main memory
  - Loses instructions and data when power off
- Non-volatile secondary memory
  - Magnetic disk
  - Flash memory
  - Optical disk (CDROM, DVD)

Networks

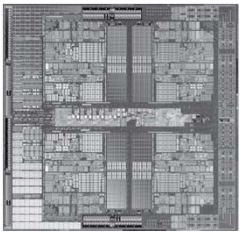
- Communication and resource sharing
- Local area network (LAN): Ethernet
  - Within a building
- Wide area network (WAN: the Internet
- Wireless network: WiFi, Bluetooth

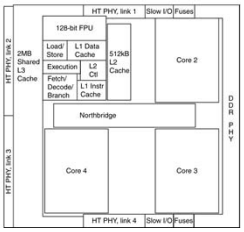

Inside the Processor (CPU)

- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory
  - Small fast SRAM memory for immediate access to data

Inside the Processor

- AMD Barcelona: 4 processor cores





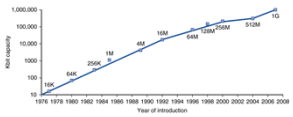
Abstractions

The BIG Picture

- Abstraction helps us deal with complexity
  - Hide lower-level detail
- Instruction set architecture (ISA)
  - The hardware/software interface
- Application binary interface
  - The ISA plus system software interface
- Implementation
  - The details underlying and interface

Technology Trends

- Electronics technology continues to evolve
  - Increased capacity and performance
  - Reduced cost



DRAM capacity

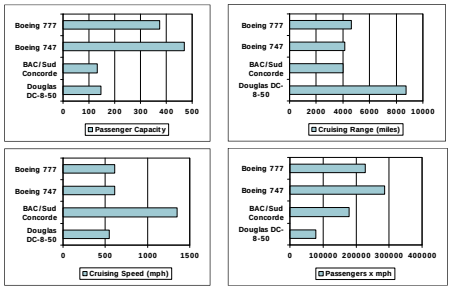
Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2005	Ultra large scale IC	6,200,000,000

Performance

- Which computer is better? Even which computer is faster?
- We can not answer the question without defining what we mean by "*better*" or "*faster*"

Defining Performance

- Which airplane has the best performance?



Response Time and Throughput

- Response time
  - How long it takes to do a task
- Throughput
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour
- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?
- We'll focus on response time for now…

Relative Performance

- Define Performance = 1/Execution Time
- "X is n time faster than Y"

$$\text{Performance}_X / \text{Performance}_Y = \text{Execution time}_Y / \text{Execution time}_X = n$$

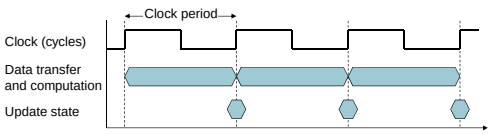
- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time_B / Execution Time_A = 15s / 10s = 1.5
  - So A is 1.5 times faster than B (***speedup***)

Measuring Execution Time

- Elapsed time
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance
- CPU time
  - Time spent processing a given job
    - Discounts I/O time, other jobs' shares
  - Comprises user CPU time and system CPU time
  - Different programs are affected differently by CPU and system performance

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock



  - Clock period: duration of a clock cycle
    - e.g., 250ps = 0.25ns = 250×10^{-12}s
  - Clock frequency (rate): cycles per second
    - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

CPU Time

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$
$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock be?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$
$$\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$$
$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$
$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

Instruction Count and CPI

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$
$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$
$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count for a program
  - Determined by program, ISA and compiler
- Average cycles per instruction
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$
$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$ ← A is faster…

$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$
$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$ ← …by this much

CPI in More Detail

- If different instruction classes take different numbers of cycles

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

- Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \frac{\text{Instruction Count}_i}{\text{Instruction Count}} \right)$$

Relative frequency

CPI Example

- Alternative compiled code sequences using instructions in classes A, B, C

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1: IC = 5
- Clock Cycles
  = 2×1 + 1×2 + 2×3
  = 10
- Avg. CPI = 10/5 = 2.0

- Sequence 2: IC = 6
- Clock Cycles
  = 4×1 + 1×2 + 1×3
  = 9
- Avg. CPI = 9/6 = 1.5

Performance Summary

The BIG Picture

$$\text{CPU Time} = \underbrace{\frac{\text{Instructions}}{\text{Program}}}_{\text{IC}} \times \underbrace{\frac{\text{Clock cycles}}{\text{Instruction}}}_{\text{CPI}} \times \underbrace{\frac{\text{Seconds}}{\text{Clock cycle}}}_{T_C}$$

- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture: affects IC, CPI, T_c

Benchmarks

- Performance is not determined by measuring the performance of a single program.
- A bunch of programs to measure performance
- SPEC CPU for example
- Spec rating is geometric mean of performance (1/execution_time)

Reporting Performance

- Assume 3programs and 3 systems

	P1	P2	P3
A	10	8	25
B	12	9	20
C	8	8	30

- Arithmetic mean
- Geometric mean

Reporting Performance

- 2 Programs and 3 machines

	A	B	C
P1	1	10	20
P2	1000	100	20

SPEC Cfp2000

Bencmark	Ultra5 t	Optero n	EPECR atio	Itanium 2	SPECR atio	O/I time	O/ISpe cR
wupwise	1600	51.5	31.06	56.1	28.53	0.92	0.92
Swim	3100	125	24.73	70.7	43.85	1.77	1.77
Mgrid	1800	98	18.37	65.8	27.36	1.49	1.49
Applu	2100	94	22.43	50.9	41.25	1.85	1.85
Mesa	1400	64.6	21.69	108.0	12.99	0.6	0.6
Galgel	2900	86.4	33.57	40.0	72.47	2.16	2.16
Art	2600	92	28.13	21.0	123.67	4.40	4.40
Equake	1300	72.6	17.92	36.3	35.78	2.00	2.00
Facerec	1900	73.6	25.80	86.9	21.86	0.85	0.85
Ammp	2200	136	16.14	132.0	16.63	1.03	1.03
Lucas	2000	88.8	22.52	107.0	18.76	0.83	0.83
Fma3d	2100	120	17.48	131.0	16.09	0.92	0.92
Sixtrack	1100	123	8.95	68.8	15.99	1.79	1.79
Aspi	2600	150	17.36	231.0	11.27	0.65	0.65
GM			**20.86**		**27.12**	**1.30**	**1.30**