香 港 中 文 大 學
The Chinese University of Hong Kong

CENG3420

Lecture 01: Introduction

Bei Yu

byu@cse.cuhk.edu.hk

(Latest update: January 10, 2018)

Fall 2017

Overview

Overview

Course Administration

Instructor:

- Bei Yu (byu@cse.cuhk.edu.hk)
- Office: SHB 914
- Office Hrs: H13:30–15:30

Tutors:

- Haoyu Yang (hyyang@cse.cuhk.edu.hk)
- Tinghuan Chen (thchen@cse.cuhk.edu.hk)
- Office: SHB 913

Grading Information

Grade Determinates

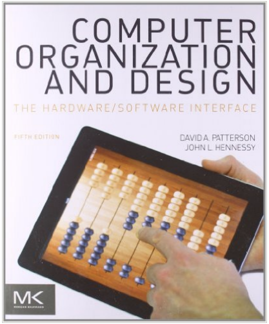
5%	Attendance
15%	Homework
15%	Midterm (Mar. 16)
25%	Three Labs (Individual project)
40%	Final Exam

- Late submission **per day** is subject to 10% of penalty.
- A student must gain at least 50% of the full marks in order to pass the course.
- A student must attend at least 80% of lectures in order to gain all class attendance credits.

General References

Textbook:

- *Computer Organization and Design*, 5th Edition
- Soft copy, `amazon.cn`, or `amazon.com`


COMPUTER ORGANIZATION AND DESIGN
THE HARDWARE/SOFTWARE INTERFACE
FIFTH EDITION
DAVID A. PATTERSON
JOHN L. HENNESSY
MK

Manuals:

- LC-3 Instruction Set Architecture (ISA)
- Lab tutorials (slides)

Slides:

- On the course web page before lecture
- Summary may be uploaded afterwards

Course Content

- Introduction to the major components of a computer system, how they function together in executing a program.
- Introduction to CPU datapath and control unit design
- Introduction to techniques to improve performance and energy-efficiency of computer systems
- Introduction to multiprocessor architecture

Course Content

- Introduction to the major components of a computer system, how they function together in executing a program.
- Introduction to CPU datapath and control unit design
- Introduction to techniques to improve performance and energy-efficiency of computer systems
- Introduction to multiprocessor architecture

Philosophy

To learn what determines the capabilities and performance of computer systems and to understand the interactions between the computer's architecture and its software so that future software designers (compiler writers, operating system designers, database programmers, application programmers, ...) can achieve the best cost-performance trade-offs and so that future architects understand the effects of their design choices on software.

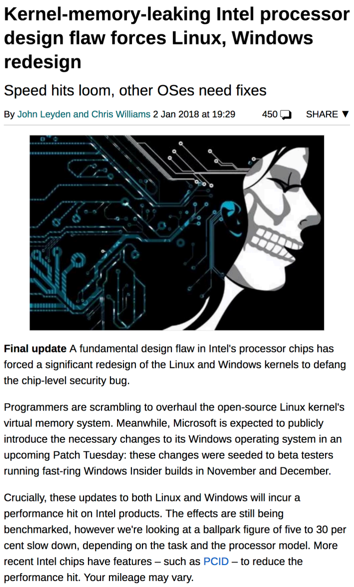
Why Learn This Stuff?

- You want to call yourself a “computer scientist/engineer”
- You want to build HW/SW people use (so need performance/power)
- You need to make a purchasing decision or offer “expert” advice

Both hardware and software affect performance/power

- Algorithm determines number of source-level statements
- Language/compiler/architecture determine the number of machine-level instructions
- Processor/memory determine how fast and how power-hungry machine-level instructions are executed

Kernel-memory-leaking Intel Processor Design Flaw



Kernel-memory-leaking Intel processor design flaw forces Linux, Windows redesign

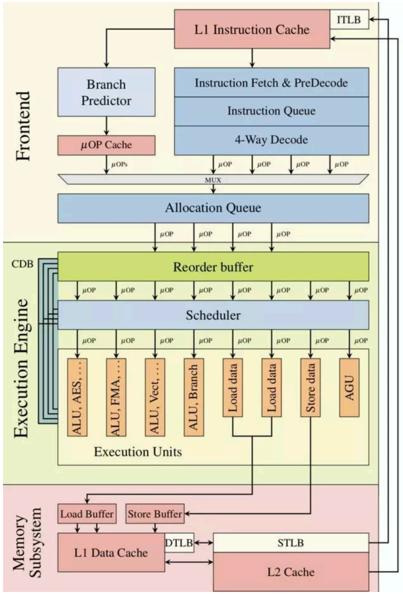
Speed hits loom, other OSes need fixes

By John Leyden and Chris Williams 2 Jan 2018 at 19:29 450 SHARE ▼

Final update A fundamental design flaw in Intel's processor chips has forced a significant redesign of the Linux and Windows kernels to defang the chip-level security bug.

Programmers are scrambling to overhaul the open-source Linux kernel's virtual memory system. Meanwhile, Microsoft is expected to publicly introduce the necessary changes to its Windows operating system in an upcoming Patch Tuesday: these changes were seeded to beta testers running fast-ring Windows Insider builds in November and December.

Crucially, these updates to both Linux and Windows will incur a performance hit on Intel products. The effects are still being benchmarked, however we're looking at a ballpark figure of five to 30 per cent slow down, depending on the task and the processor model. More recent Intel chips have features – such as PCID – to reduce the performance hit. Your mileage may vary.



http://www.theregister.co.uk/2018/01/02/intel_cpu_design_flaw/

What You Should Already Know

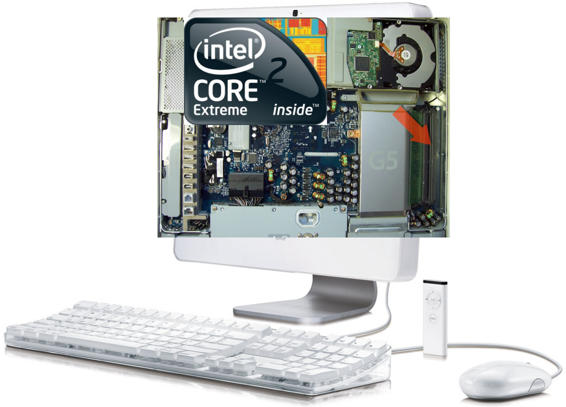
- Basic logic design & machine organization
  - logical minimization, FSMs, component design
  - processor, memory, I/O
- Create, run, debug programs in an assembly language
  - Will be introduced in tutorial
- Create, compile, and run C/C++ programs
- Create, organize, and edit files and run programs on Unix/Linux

Computer Organization and Design

- This course is all about how computers work
- But what do we mean by a computer?
  - Different types: embedded, laptop, desktop, server
  - Different uses: automobiles, graphics, finance, genomics ...
  - Different manufacturers: Intel, Apple, IBM, Sony, Oracle ...
  - Different underlying technologies and different costs
- Analogy: Consider a course on “automotive vehicles”
  - Many similarities from vehicle to vehicle (e.g., wheels)
  - Huge differences from vehicle to vehicle (e.g., gas vs. electric)
- **Best way to learn**:
  - Focus on a specific instance and learn how it works
  - While learning general principles and historical perspectives

Overview

A Computer



zZounds

Desktop computers

Designed to deliver good performance to a single user at low cost usually executing 3rd party software, usually incorporating a graphics display, a keyboard, and a mouse

Other Classes of Computers

Servers

Used to run larger programs for multiple, simultaneous users typically accessed only via a network and that places a greater emphasis on dependability and (often) security

Supercomputers

A high performance, high cost class of servers with hundreds to thousands of processors, terabytes of memory and petabytes of storage that are used for high-end scientific and engineering applications.

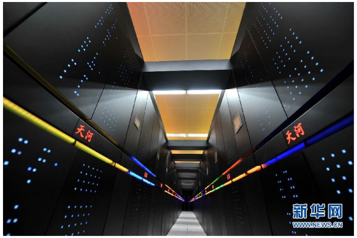
Embedded computers (processors)

A computer inside another device used for running one predetermined application

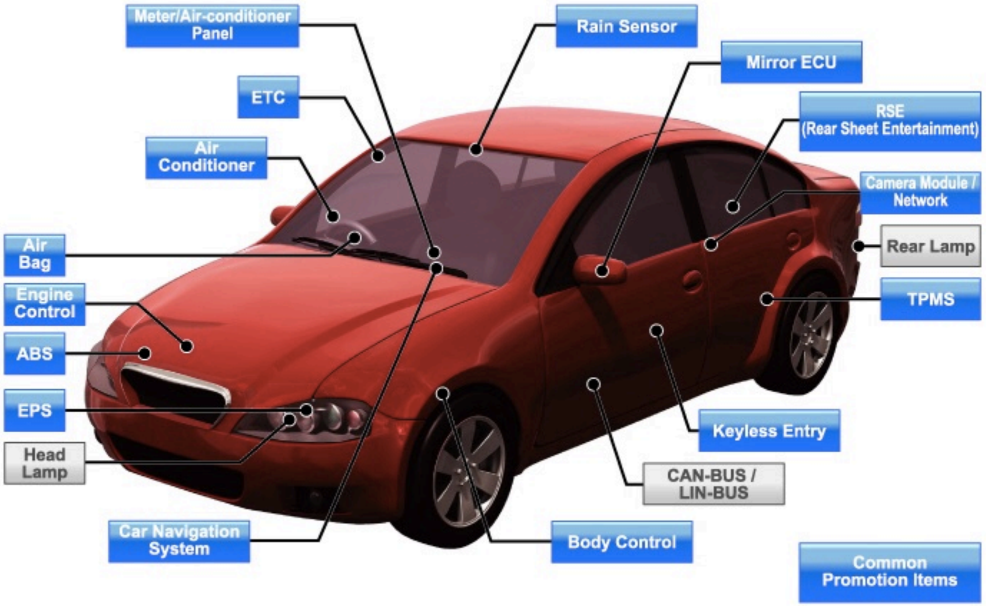
Supercomputers

Tianhe-2 (MilkyWay-2)

- Over 3 million cores
- Power: 17.6 MW (24 MW with cooling)
- Speed: 33.86 PFLOPS (peta $= 10^{15}$)



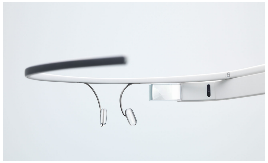
Embedded Computers in You Car



PostPC Era

Personal Mobile Device (PMD)

Battery-operated device with wireless connectivity

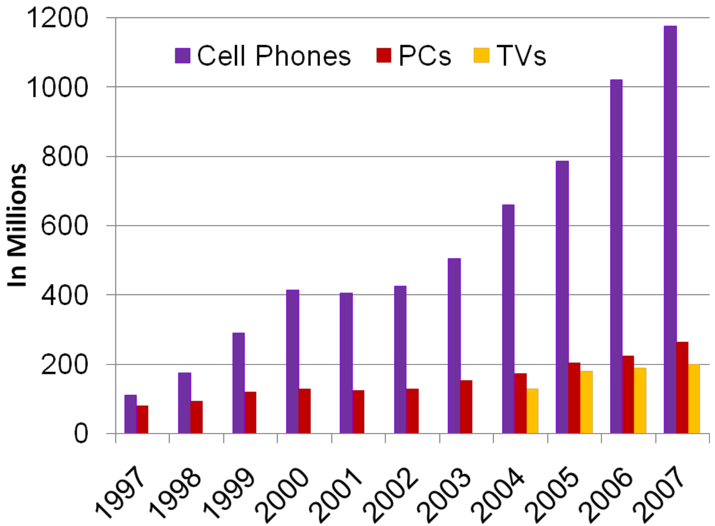


Warehouse Scale Computer (WSC)

Datacenter containing hundreds of thousands of servers providing software as a service (**SaaS**)

Growth in Cell Phone Sales (Embedded)

- embedded growth $>>$ desktop growth
- Where else are embedded processors found?



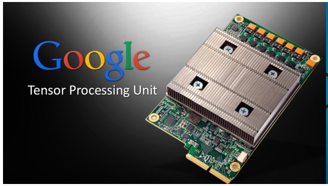
When Machine Learning Meets Hardware

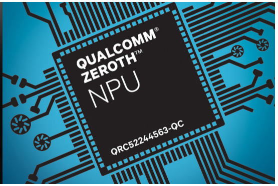
Convolution layer is one of the most expensive layers

- Computation pattern
- Emerging challenges

More and more end-point devices with limited memory

- Cameras
- Smartphone
- Autonomous driving





DEEPHI TECH

MOBILEYE®
An intel company

Convolutional Neural Network (CNN)

Autonomous drive

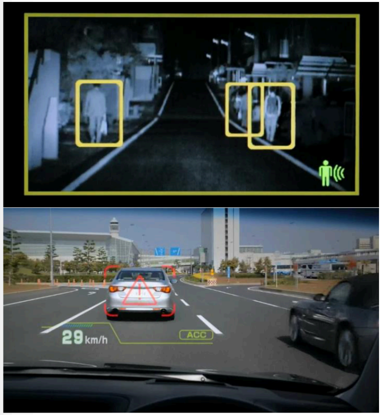
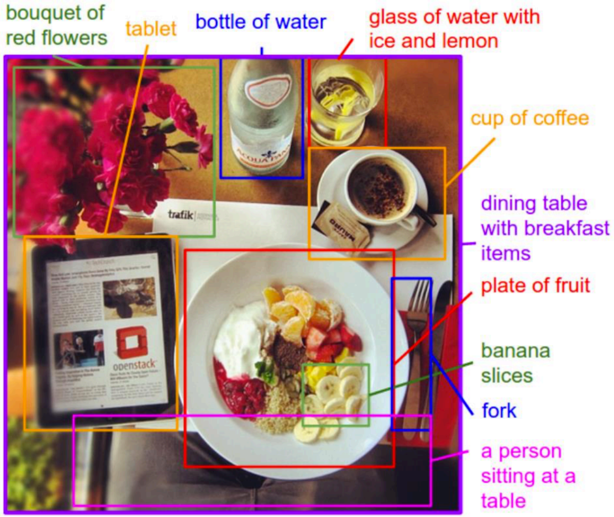
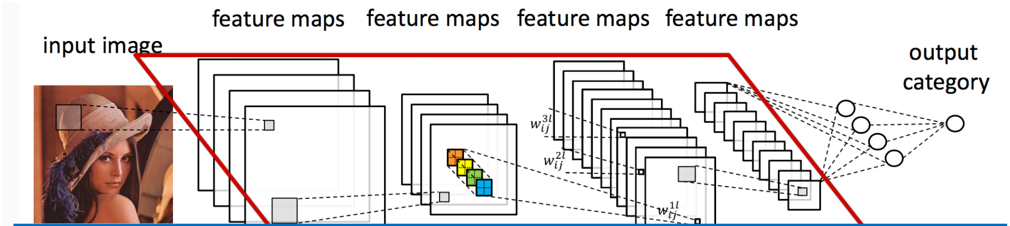


Image recognition

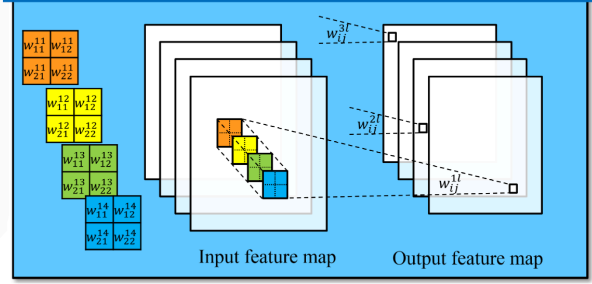


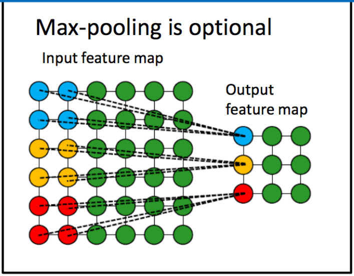
Bottleneck of CNN



Convolutional layers account for over 90% computation

[1] A. Krizhevsky, etc. Imagenet classification with deep convolutional neural networks. NIPS 2012.
[2] J. Cong and B. Xiao. Minimizing computation in convolutional neural networks. ICANN 2014


w_{11}^{11} w_{12}^{11} w_{21}^{11} w_{22}^{11}
w_{11}^{12} w_{12}^{12} w_{21}^{12} w_{22}^{12}
w_{11}^{13} w_{12}^{13} w_{21}^{13} w_{22}^{13}
w_{11}^{14} w_{12}^{14} w_{21}^{14} w_{22}^{14}
w_{ij}^{3l}
w_{ij}^{2l}
w_{ij}^{1l}
Input feature map
Output feature map



In-Datacenter Performance Analysis of a Tensor Processing Unit™

Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon

Google, Inc., Mountain View, CA USA

Email: {jouppi, cliffy, nishantpatil, davidpatterson} @google.com

To appear at the 44th International Symposium on Computer Architecture (ISCA), Toronto, Canada, June 26, 2017.

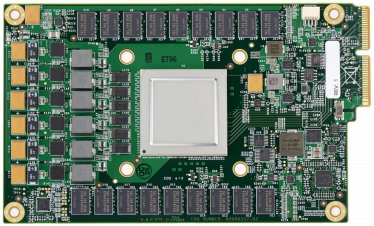


Figure 3. TPU Printed Circuit Board. It can be inserted in the slot for an SATA disk in a server, but the card uses PCIe Gen3 x16.

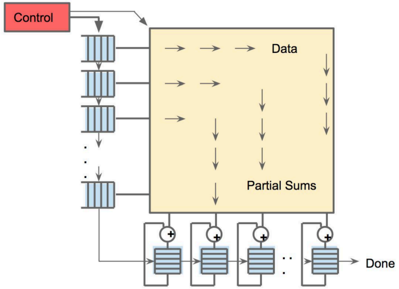
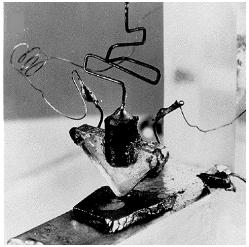


Figure 4. Systolic data flow of the Matrix Multiply Unit. Software has the illusion that each 256B input is read at once, and they instantly update one location of each of 256 accumulator RAMs.

The Evolution of Computer Hardware

When was the first transistor invented?

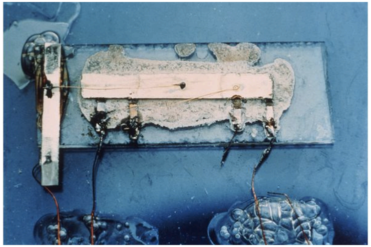

(a)

(b)

(a) 1947, bi-polar transistor, by John Bardeen et al. at Bell Laboratories; (b) UNIVAC I (Universal Automatic Computer): the first commercial computer in USA.

The Evolution of Computer Hardware

When was the first IC (integrated circuit) invented?



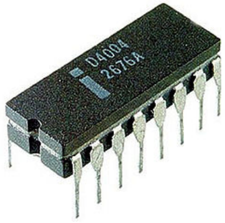
(a)

(b)

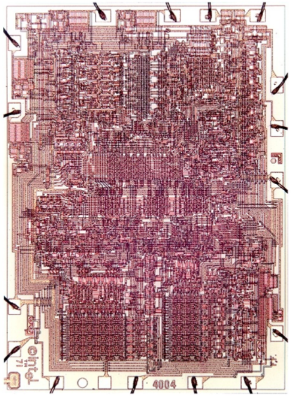
(a) 1958, by Jack Kilby@Texas Instruments, by hand. Several transistors, resistors and capacitors on a single substrate. (b) IBM System/360, 2MHz, 128KB – 256KB.

The Evolution of Computer Hardware

When was the first Microprocessor?



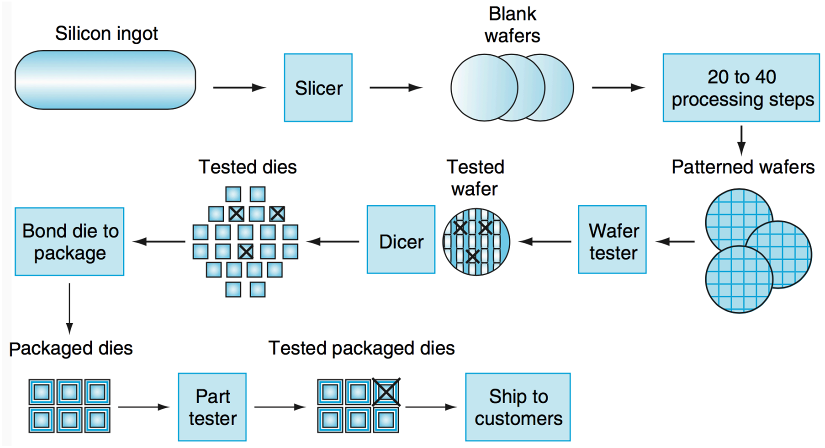
(a)



(b)

1971, Intel 4004.

The IC Manufacturing Process

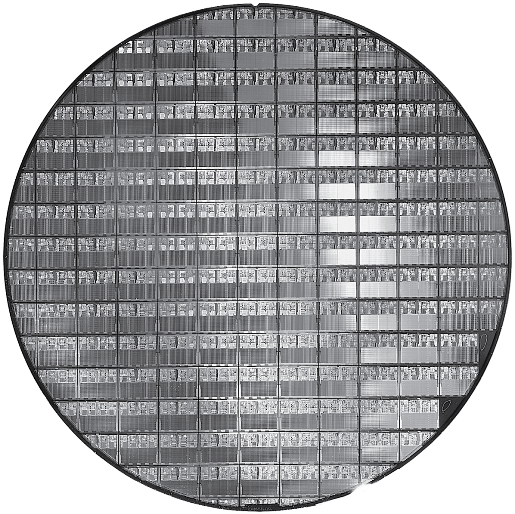


Yield

Proportion of working dies per wafer

Check this: `https://youtu.be/d9SWNLZvA8g?list=FLELqiXCJQW-jcijW8ZAbA8w`

AMD Opteron X2 Wafer



300mm wafer, 117 chips, 90nm technology.

Integrated Circuit Cost

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \cdot \text{Yield}}$$

$$\text{Dies per wafer} = \text{Wafer area / Die area}$$

$$\text{Yield} = \frac{1}{[1 + (\text{Defects per area} \cdot \text{Die area / 2})]^2}$$

Nonlinear relation to area and defect rate

- Wafer cost and area are fixed
- Defect rate determined by manufacturing process
- Die area determined by architecture and circuit design

Impacts of Advancing Technology

Processor

- Logic capacity: increases about 30% per year
- Performance: 2$\times$ every 1.5 years

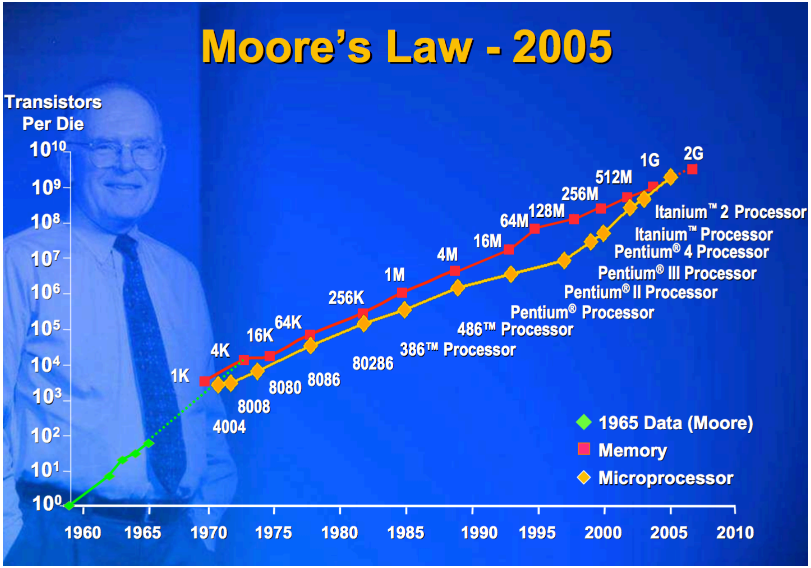
Memory

- DRAM capacity: 4$\times$ every 3 years, about 60% per year
- Memory speed: 1.5$\times$ every 10 years
- Cost per bit: decreases about 25% per year

Disk

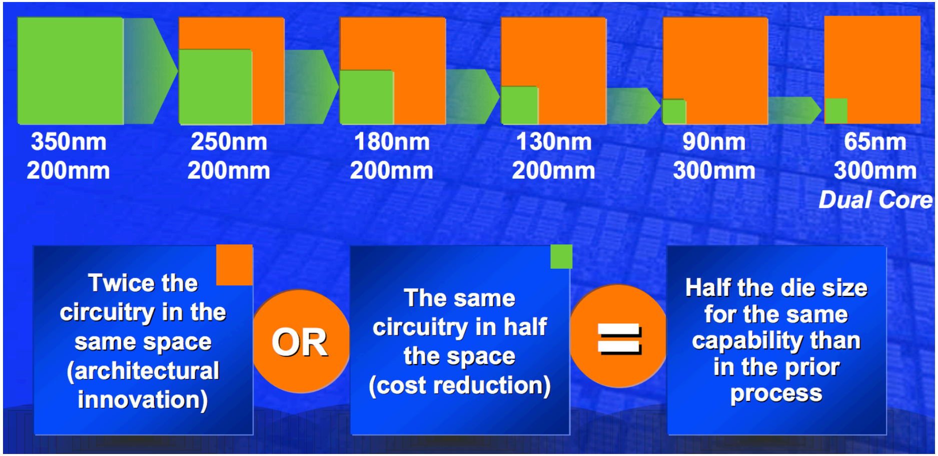
- Capacity: increases about 60% per year

Moore’s Law for CPUs and DRAMs



From: “*Facing the Hot Chips Challenge Again*”, Bill Holt, Intel, presented at Hot Chips 17, 2005.

Main driver: device scaling ...



350nm 200mm
250nm 200mm
180nm 200mm
130nm 200mm
90nm 300mm
65nm 300mm *Dual Core*
Twice the circuitry in the same space (architectural innovation)
OR
The same circuitry in half the space (cost reduction)
=
Half the die size for the same capability than in the prior process

From: "*Facing the Hot Chips Challenge Again*", Bill Holt, Intel, presented at Hot Chips 17, 2005.

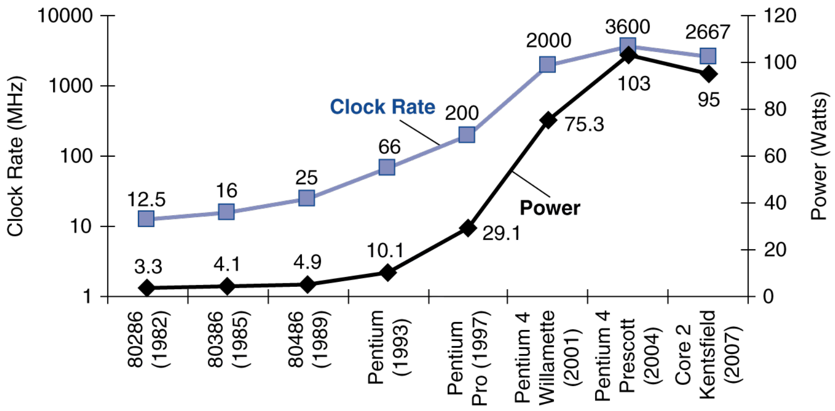
Technology Scaling Road Map (ITRS)

Year	2004	2006	2008	2010	2012
Feature size (nm)	90	65	45	32	22
Intg. Capacity (BT)	2	4	6	16	32

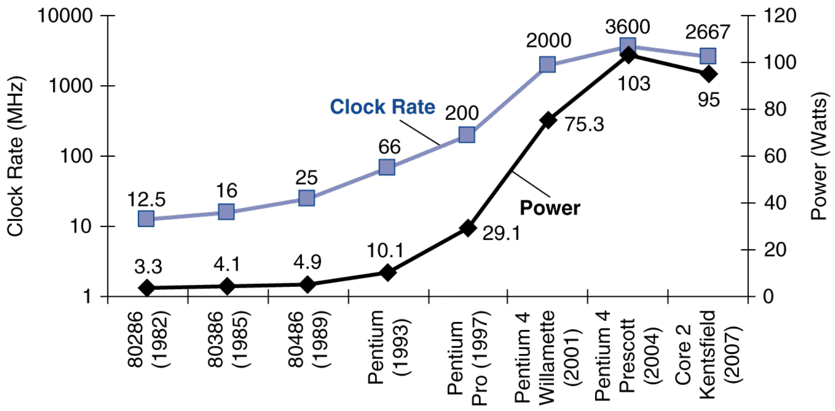
Fun facts about 45nm transistors

- 30 million can fit on the head of a pin
- You could fit more than 2,000 across the width of a human hair
- If car prices had fallen at the same rate as the price of a single transistor since 1968, a new car today would cost about 1 cent

Highest Clock Rate of Intel Processors



Highest Clock Rate of Intel Processors



What if the exponential increase had kept up? Why not?

- Due to process improvements
- Deeper pipeline
- Circuit design techniques

Power Issue

$$\text{Power} = \text{Capacitive load} \cdot \text{Voltage}^2 \cdot \text{Frequency}*$$

Example

For a simple processor, if capacitive load is reduced by 15%, voltage is reduced by 15%, maintain the same frequency, how much power consumption can be reduced?

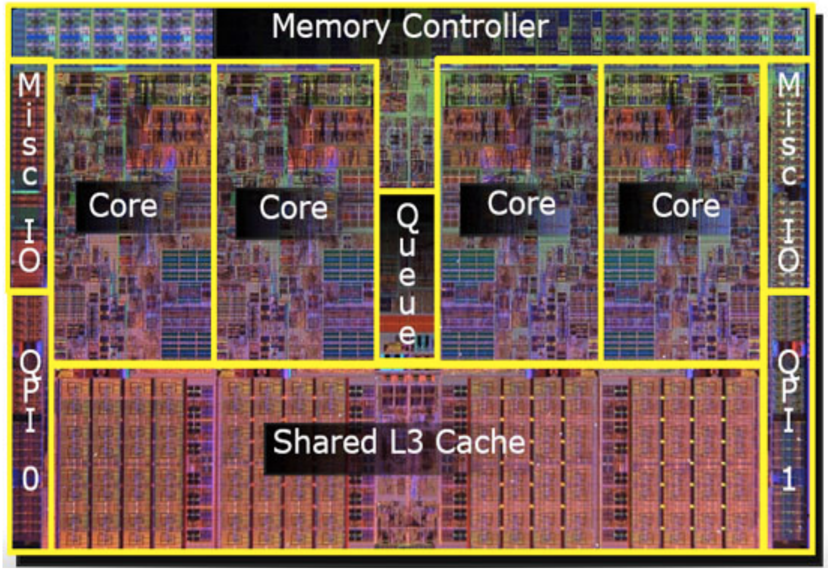
$*$here we only consider dynamic power, but not static power

A Sea Change Is at Hand

- The power challenge has forced a change in the design of microprocessors
- Since 2002 the rate of improvement in the response time of programs on desktop computers has slowed from a factor of 1.5 per year to less than a factor of 1.2 per year
- As of 2006 all desktop and server companies are shipping microprocessors with multiple processors – cores – per chip
- Plan of record is to add two cores per chip per generation (about every two years)

Product	AMD Barcelona	Intel Nehalem	IBM Power 6	Sun Niagara 2
Cores per chip	4	4	2	8
Clock rate	~2.5 GHz	~2.5 GHz	4.7 GHz	1.4 GHz
Power	120 W	~100 W	~100 W	94 W

Intel Core i7 Processor



45nm technology, 18.9mm x 13.6mm, 0.73billion transistors, 2008

Overview

What is a Computer?

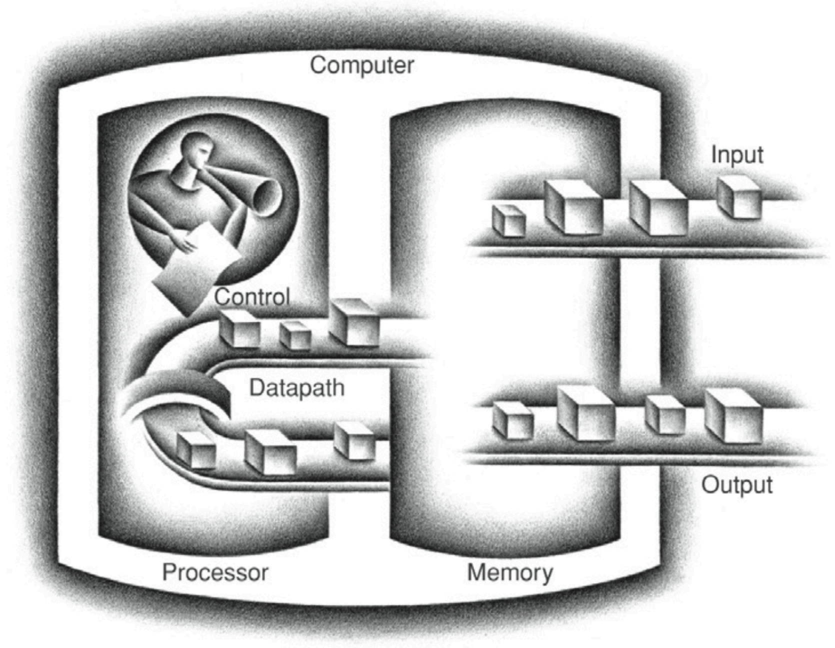
Components

- processor (datapath, control)
- input (mouse, keyboard)
- output (display, printer)
- memory (cache, main memory, disk drive, CD/DVD)
- network

Our primary focus: the processor (datapath and control) and its interaction with memory systems

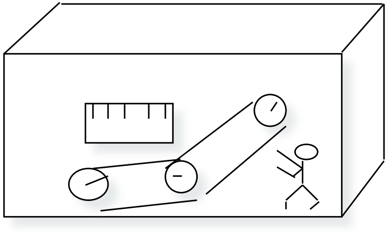
- Implemented using tens/hundreds of millions of transistors
- Impossible to understand by looking at each transistor
- We need abstraction!

Major Components of a Computer



Machine Organization

- Capabilities and performance characteristics of the principal Functional Units (FUs). (e.g., register file, ALU, multiplexors, memories, ...)
- The ways those FUs are interconnected (e.g., buses)
- Logic and means by which information flow between FUs is controlled
- The machine's Instruction Set Architecture (ISA)
- Register Transfer Level (RTL) machine description



Processor Organization

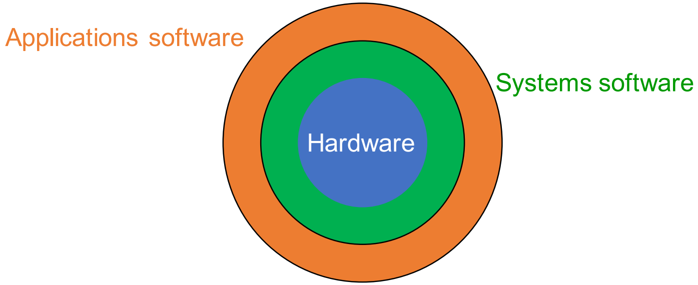
Control needs to have circuitry to

- Decide which is the next instruction and input it from memory
- Decode the instruction
- Issue signals that control the way information flows between datapath components
- Control what operations the datapath's functional units perform

Datapath needs to have circuitry to

- Execute instructions - functional units (e.g., adder) and storage locations (e.g., register file)
- Interconnect the functional units so that the instructions can be executed as required
- Load data from and store data to memory

System Software



Operating System

- Supervising program that interfaces the user's program with the hardware (e.g., Linux, iOS, Windows)
- Handles basic input and output operations
- Allocates storage and memory
- Provides for protected sharing among multiple applications

Compiler

- Translate programs written in a high-level language (e.g., C, Java) into instructions that the hardware can execute

Advantages of Higher-Level Languages ?

- Allow the programmer to think in a more natural language and for their intended use (Fortran for scientific computation, Cobol for business programming, Lisp for symbol manipulation, Java for web programming, ...)
- Improve programmer productivity – more understandable code that is easier to debug and validate
- Improve program maintainability
- Allow programs to be independent of the computer on which they are developed (compilers and assemblers can translate high-level language programs to the binary instructions of any machine)
- Emergence of optimizing compilers that produce very efficient assembly code optimized for the target machine

As a result, very little programming is done today at the assembler level

Below the Program

- High-level language program (in C)

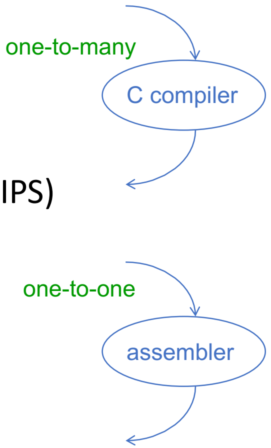
```
swap (int v[], int k)
(int temp;
        temp = v[k];
        v[k] = v[k+1];
        v[k+1] = temp;
)
```

- Assembly language program (for MIPS)

```
swap:   sll     $2, $5, 2
        add     $2, $4, $2
        lw      $15, 0($2)
        lw      $16, 4($2)
        sw      $16, 0($2)
        sw      $15, 4($2)
        jr      $31
```

- Machine (object) code (for MIPS)

```
000000 00000 00101 0001000010000000
000000 00100 00010 0001000000100000
 . . .
```



Below the Program

- High-level language program (in C)

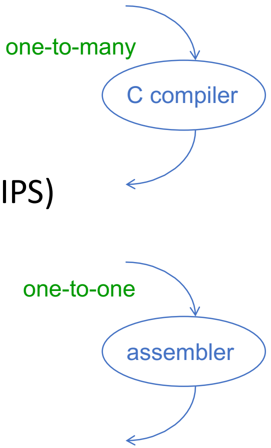
```
swap (int v[], int k)
(int temp;
        temp = v[k];
        v[k] = v[k+1];
        v[k+1] = temp;
)
```

- Assembly language program (for MIPS)

```
swap:   sll     $2, $5, 2
        add     $2, $4, $2
        lw      $15, 0($2)
        lw      $16, 4($2)
        sw      $16, 0($2)
        sw      $15, 4($2)
        jr      $31
```

- Machine (object) code (for MIPS)

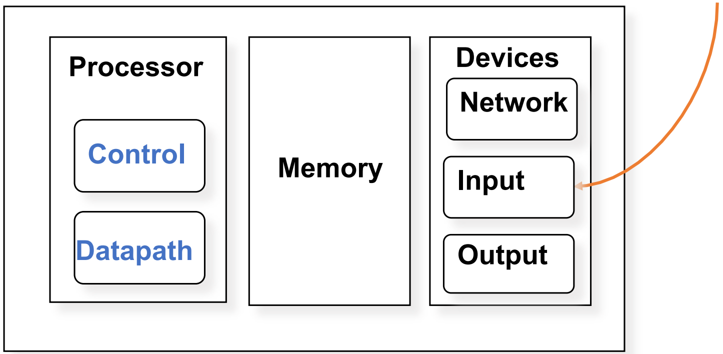
```
000000 00000 00101 0001000010000000
000000 00100 00010 0001000000100000
  .   .   .
```



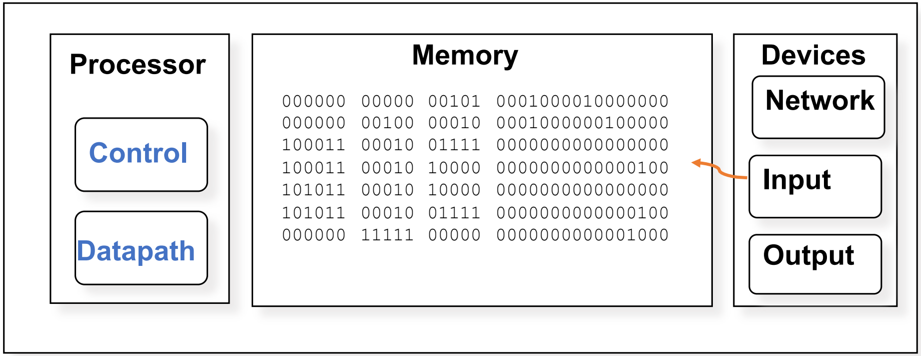
Max # of operations?

Input Device Inputs Object Code

```
000000  00000  00101  0001000010000000
000000  00100  00010  0001000000100000
100011  00010  01111  0000000000000000
100011  00010  10000  0000000000000100
101011  00010  10000  0000000000000000
101011  00010  01111  0000000000000100
000000  11111  00000  0000000000001000
```

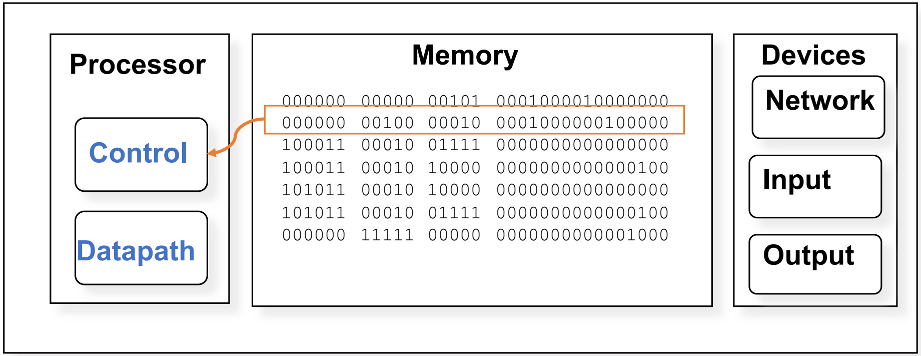


Object Code Stored in Memory



```
000000  00000  00101  0001000010000000
000000  00100  00010  0001000000100000
100011  00010  01111  0000000000000000
100011  00010  10000  0000000000000100
101011  00010  10000  0000000000000000
101011  00010  01111  0000000000000100
000000  11111  00000  0000000000001000
```

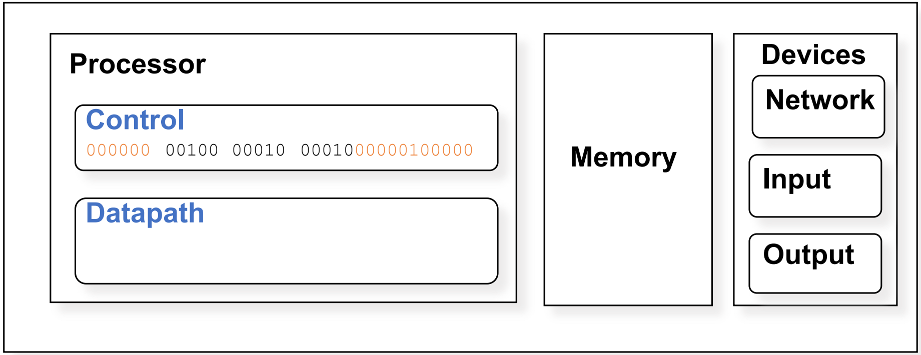
Object Code Stored in Memory



```
000000 00000 00101 0001000010000000
000000 00100 00010 0001000000100000
100011 00010 01111 0000000000000000
100011 00010 10000 0000000000000100
101011 00010 10000 0000000000000000
101011 00010 01111 0000000000000100
000000 11111 00000 0000000000001000
```

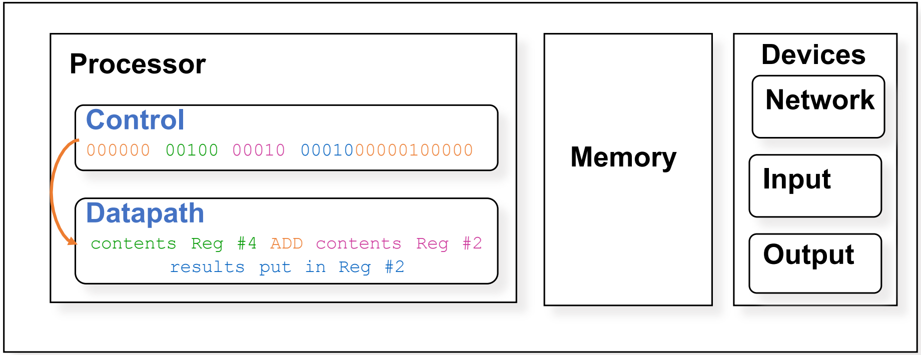
Processor fetches an instruction from memory

Decode & Excute Codes



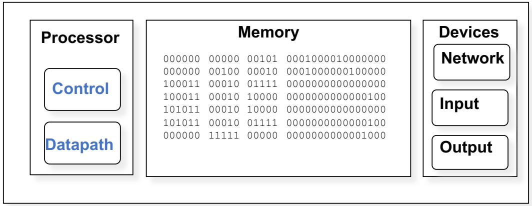
- Control decodes the instruction to determine what to execute

Decode & Excute Codes



- Control decodes the instruction to determine what to execute
- Datapath executes the instruction as directed by control

What Happens Next?

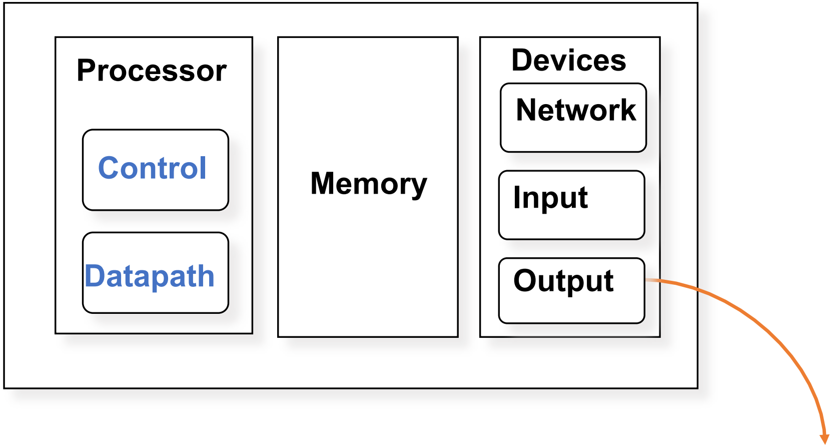


```
000000  00000  00101  0001000010000000
000000  00100  00010  0001000000100000
100011  00010  01111  0000000000000000
100011  00010  10000  0000000000000100
101011  00010  10000  0000000000000000
101011  00010  01111  0000000000000100
000000  11111  00000  0000000000001000
```

$$\begin{tikzcd} & \text{Fetch} \arrow[dr, bend left] & \\ \text{Exec} \arrow[ur, bend left] & & \text{Decode} \arrow[ll, bend left] \end{tikzcd}$$

- Processor fetches the next instruction from memory
- How does it know which location in memory to fetch from next?

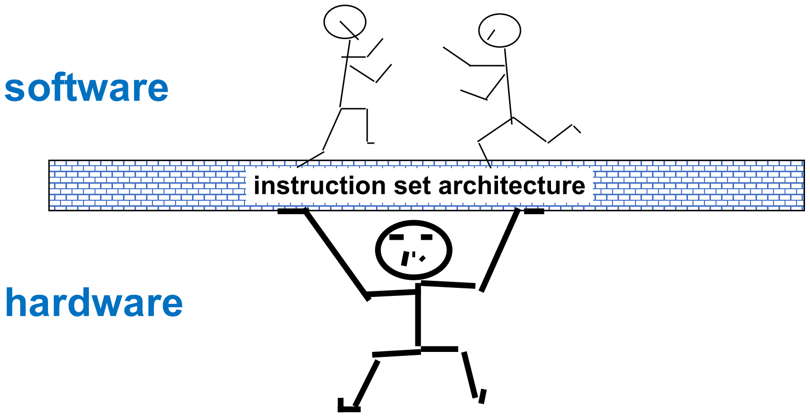
Output Device Outputs Data



```
000001000101000000000 0000 0000 000
000000000100111100000 0000 0000 100
000000111110000000000 0000 0001 000
```

Instruction Set Architecture (ISA)

The interface description separating the software and hardware



Instruction Set Architecture (ISA)

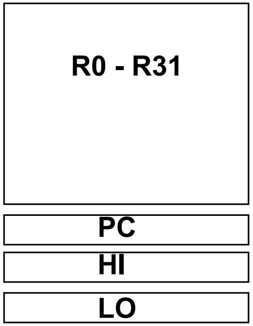
- ISA, or simply architecture – the abstract interface between the hardware and the lowest level software that includes all the information necessary to write a machine language program, including instructions, registers, memory access, I/O, ...
- Enables implementations of varying cost and performance to run identical software
- The combination of the basic instruction set (the ISA) and the operating system interface is called the application binary interface (ABI)
- ABI: The user portion of the instruction set plus the operating system interfaces used by application programmers. Defines a standard for binary portability across computers.

The MIPS ISA

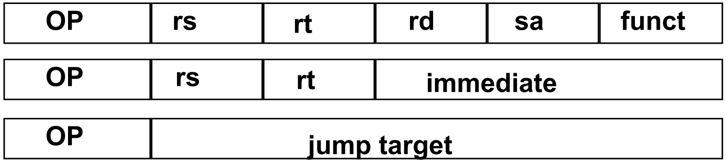
Instruction Categories

- Load/Store
- Computational
- Jump and Branch
- Floating Point
- Memory Management
- Special

Registers

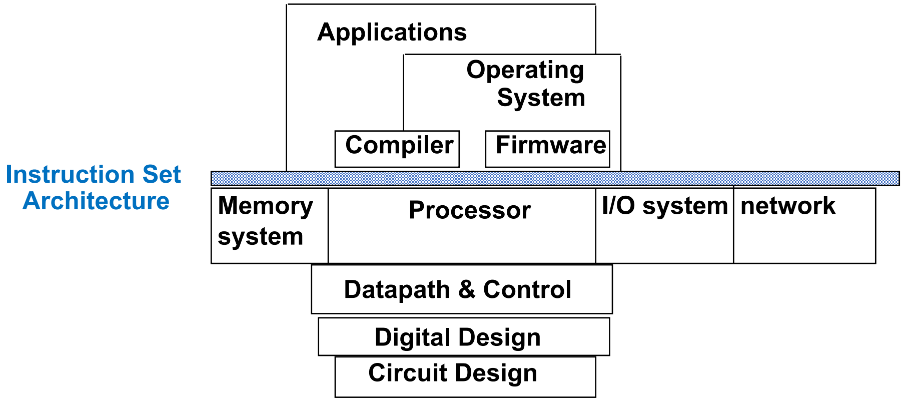


3 Instruction Formats: all 32 bits wide



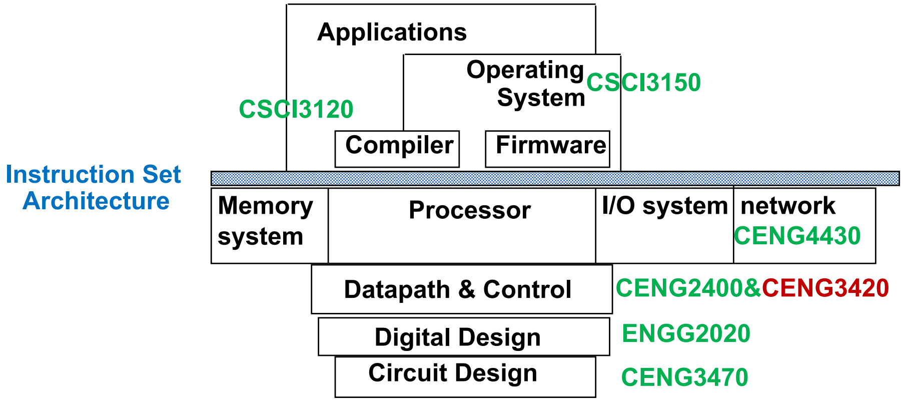
Overview

How Do the Pieces Fit Together?



- Coordination of many levels of abstraction
- Under a rapidly changing set of forces
- Design, measurement, and evaluation

How Do the Pieces Fit Together?



- Coordination of many levels of abstraction
- Under a rapidly changing set of forces
- Design, measurement, and evaluation