COMPUTER ARCHITECTURE

Winter 2023

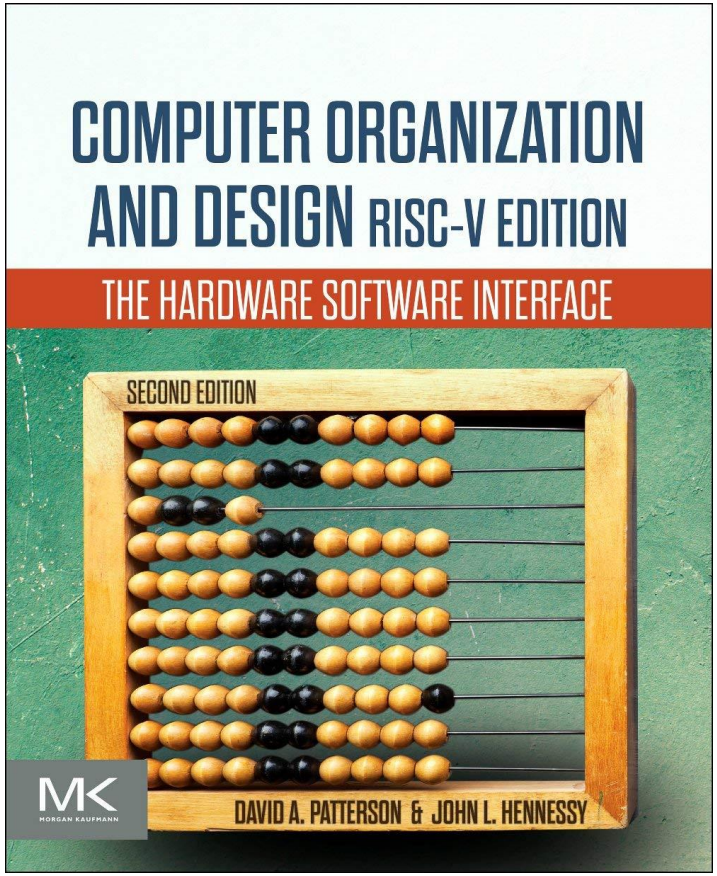
Course Organization

- Instructor: Yan Cui
  - Office hour: TBD
  - Email: ycui4@scu.edu
- Classroom: SCDI 3116, M/W 7:10-9:00
- Grader: TBD

Prerequisites

- COEN 20 (Introduction to Embedded Systems)
- COEN 21, COEN 120A (Real Time Systems) or COEN 127C (Advanced Logic Design), or equivalent.

Course Textbook



Computer Organization and Design RISC-V, **2nd** edition by Hennessy and Patterson, Morgan Kaufmann, 2020, ISBN: 978-0-12-820331-6

Grade Composition

- Homework: 10%
  - 6 assignments,
  - 3 of them will be submitted on Camino
- Project: 15%
  - Assembly code
  - Data path and Control
  - Report
  - Presentation

Grade Composition

- Quiz: 10%
  - 2 Quizzes during lecture time
- Midterm: 30%
  - The contents we learned
- Final Exam: 35%
  - Cumulative

Do You Grade with a Curve

- Grade with A curve if necessary

Camino

- Announcements
  - Important notes and reminder from me.
- Lecture slides and related files are uploaded into "Files" section
- Homework assignments and Project submissions
  - PDF is the standard file format for downloading and uploading, unless otherwise specified

Chapter 1

- Computers
- Seven Great Ideas
- Below Your program
- Under the Covers
- Performance
- The Power Wall
- From Uniprocessors to multiprocessors
- Benchmarking
- Pitfalls
- Binary and Endianness

The Computer Revolution

- Progress in computer technology
  - Domain specific architecture (DSA)
- Makes novel applications feasible
  - Computers in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search Engines

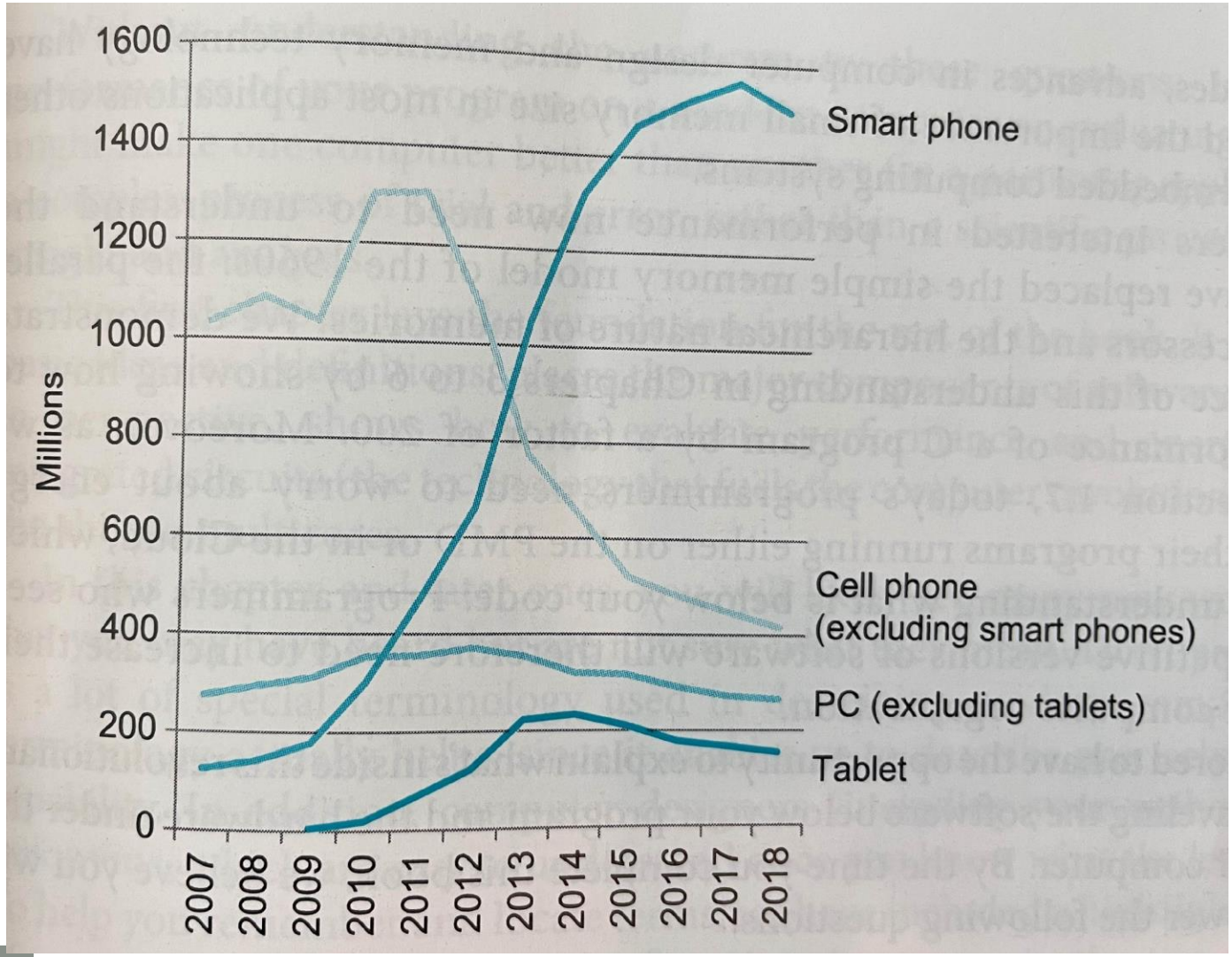
Traditional Classes of Computers

- Personal 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

Traditional Classes of Computers

- 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

The Post-PC Era



The Post-PC Era

- Personal Mobile Device (PMD)
  - Battery operated
  - Connects to the Internet
  - Hundreds of dollars
  - Smart phones, tablets, electronic glasses
- Cloud computing
  - Warehouse Scale Computers (WSC)
  - Software as a Service (SaaS)
  - Portion of software run on a PMD and
  - portion run in the Cloud
  - Amazon and Google

What You Will Learn

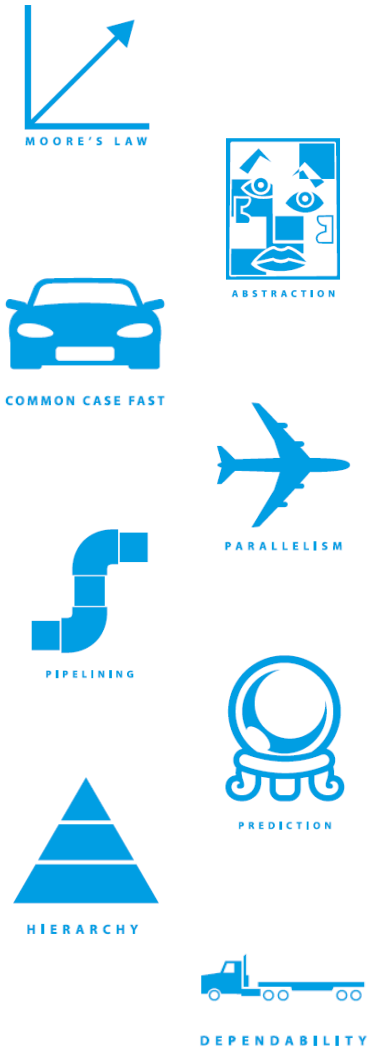
- How the hardware executes machine code
- The hardware/software interface
- What determines program performance
  - And how it can be improved
- How hardware designers improve performance

Understanding Performance

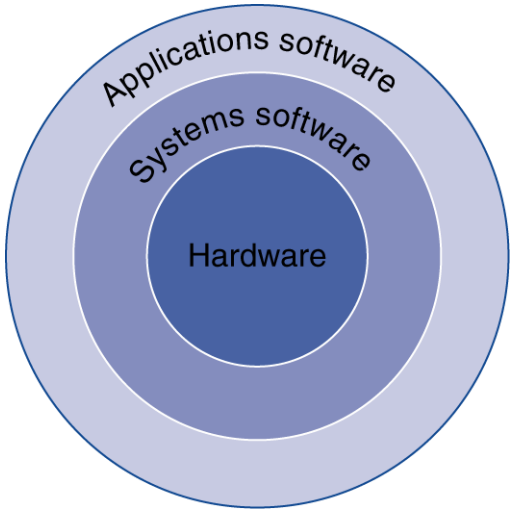
- Algorithm
  - Determines number of operations executed
- Programming language, compiler, architecture
  - Determine number of machine instructions executed per operation
- Processor and memory system
  - Determine how fast instructions are executed
- I/O system (including OS)
  - Determines how fast I/O operations are executed

Seven Great Ideas

- Use ***abstraction*** to simplify design
- Make the ***common case fast***
- Performance *via* ***parallelism***
- Performance *via* ***pipelining***
- Performance *via* ***prediction***
- ***Hierarchy*** of memories
- ***Dependability*** *via* redundancy



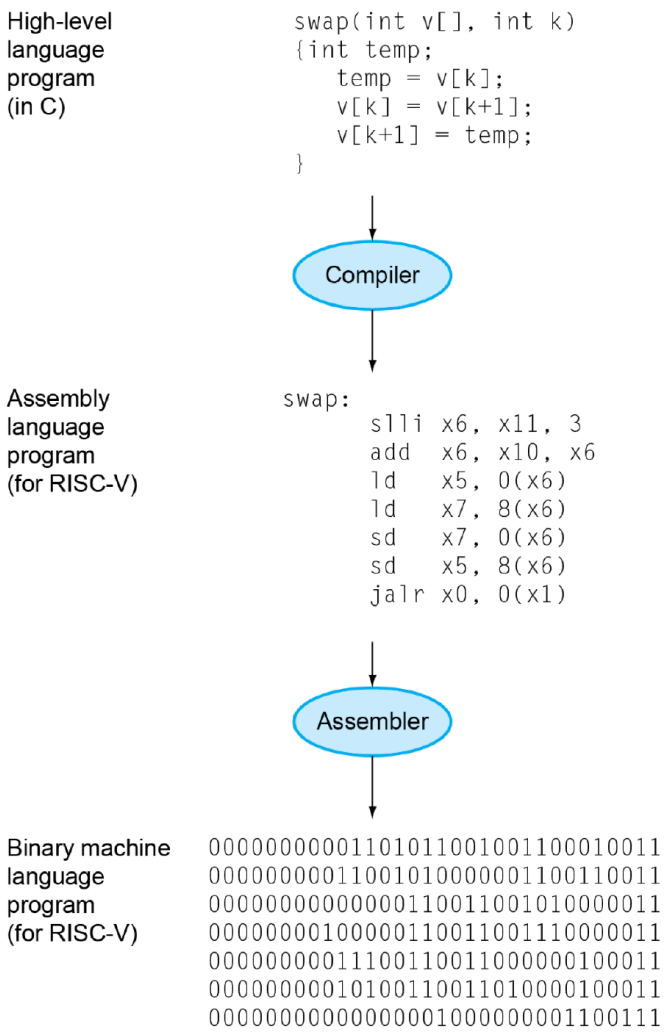
Below Your Program



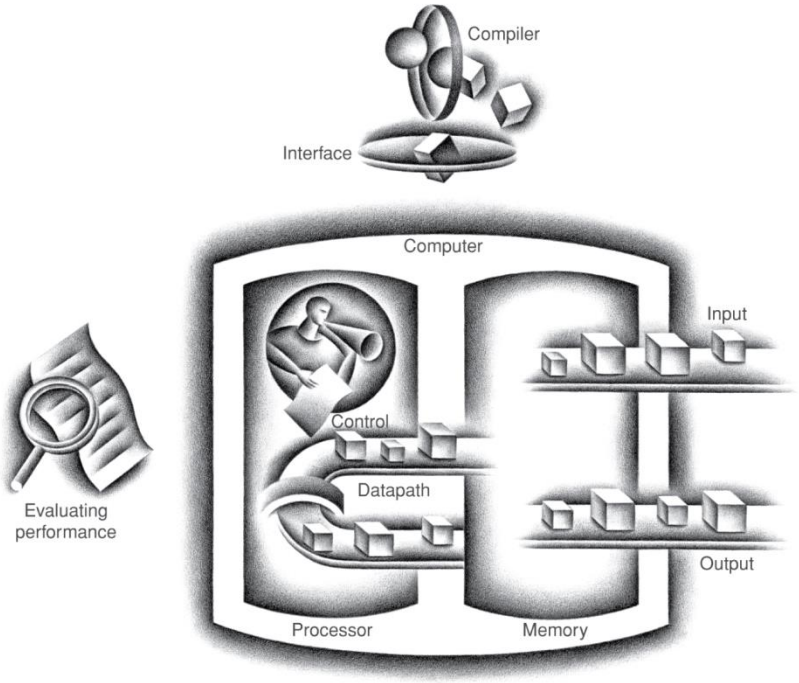
- Application software
  - Written in high-level language
- System software
  - Compiler: translates HLL code to machine code
  - Operating System: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - Processor, memory, I/O controllers

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

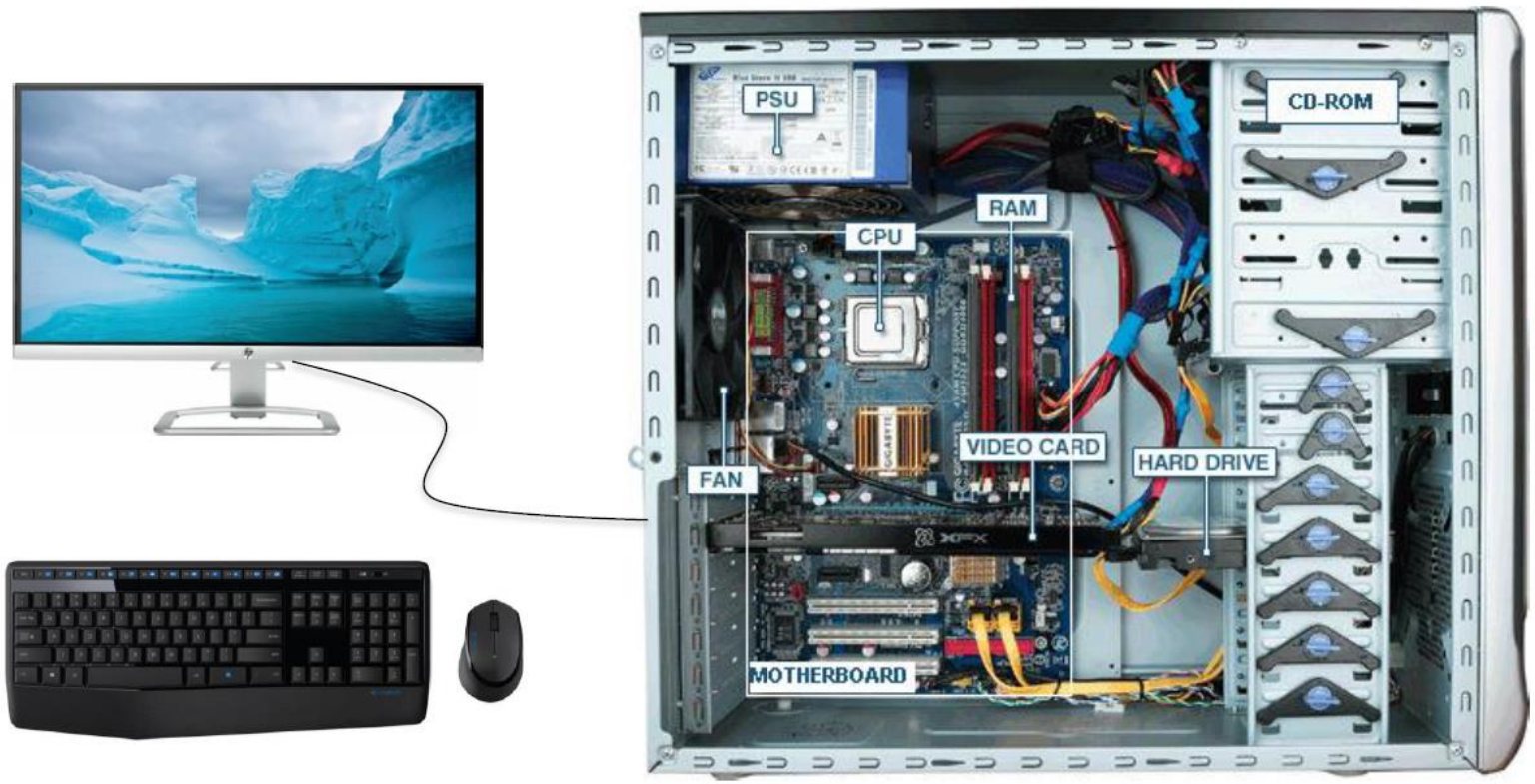


Components of a Computer



- 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

Inside of a PC



Inside of Smartphone

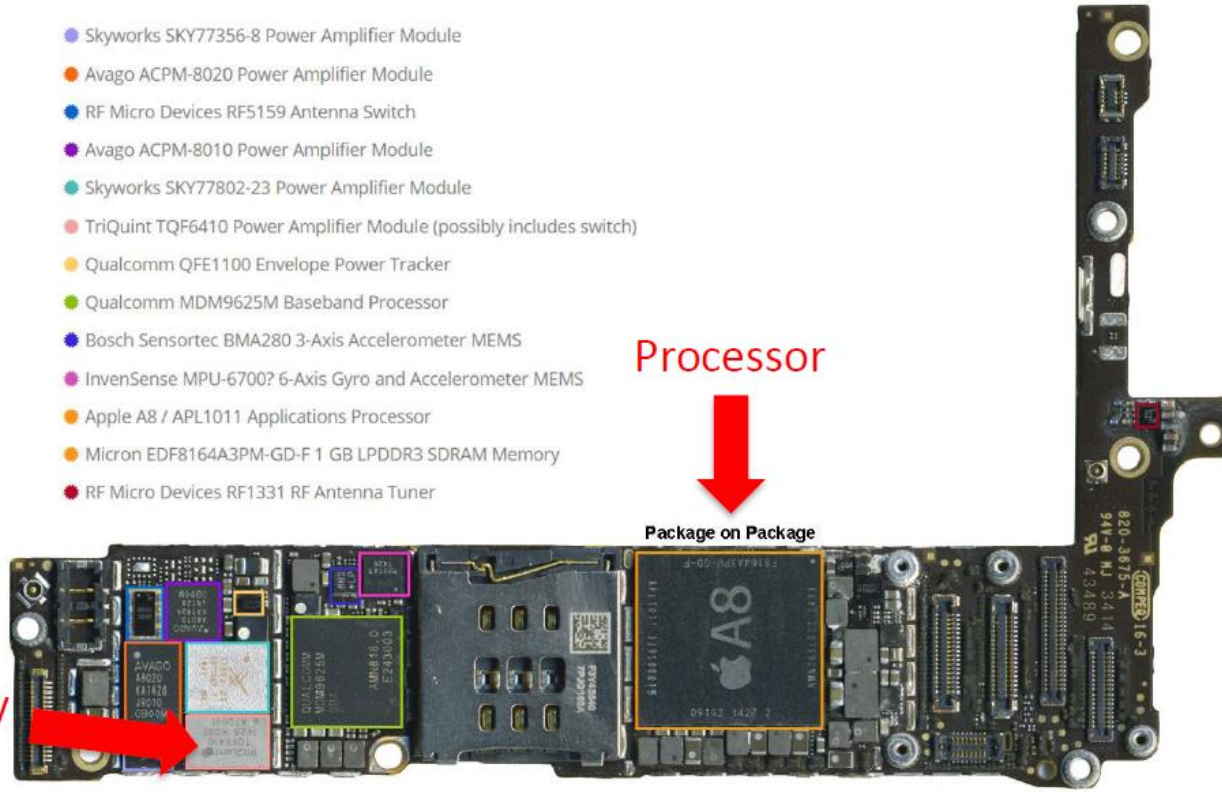
Inside of Smartphone

- Skyworks SKY77356-8 Power Amplifier Module
- Avago ACPM-8020 Power Amplifier Module
- RF Micro Devices RF5159 Antenna Switch
- Avago ACPM-8010 Power Amplifier Module
- Skyworks SKY77802-23 Power Amplifier Module
- TriQuint TQF6410 Power Amplifier Module (possibly includes switch)
- Qualcomm QFE1100 Envelope Power Tracker
- Qualcomm MDM9625M Baseband Processor
- Bosch Sensortec BMA280 3-Axis Accelerometer MEMS
- InvenSense MPU-6700? 6-Axis Gyro and Accelerometer MEMS
- Apple A8 / APL1011 Applications Processor
- Micron EDF8164A3PM-GD-F 1 GB LPDDR3 SDRAM Memory
- RF Micro Devices RF1331 RF Antenna Tuner

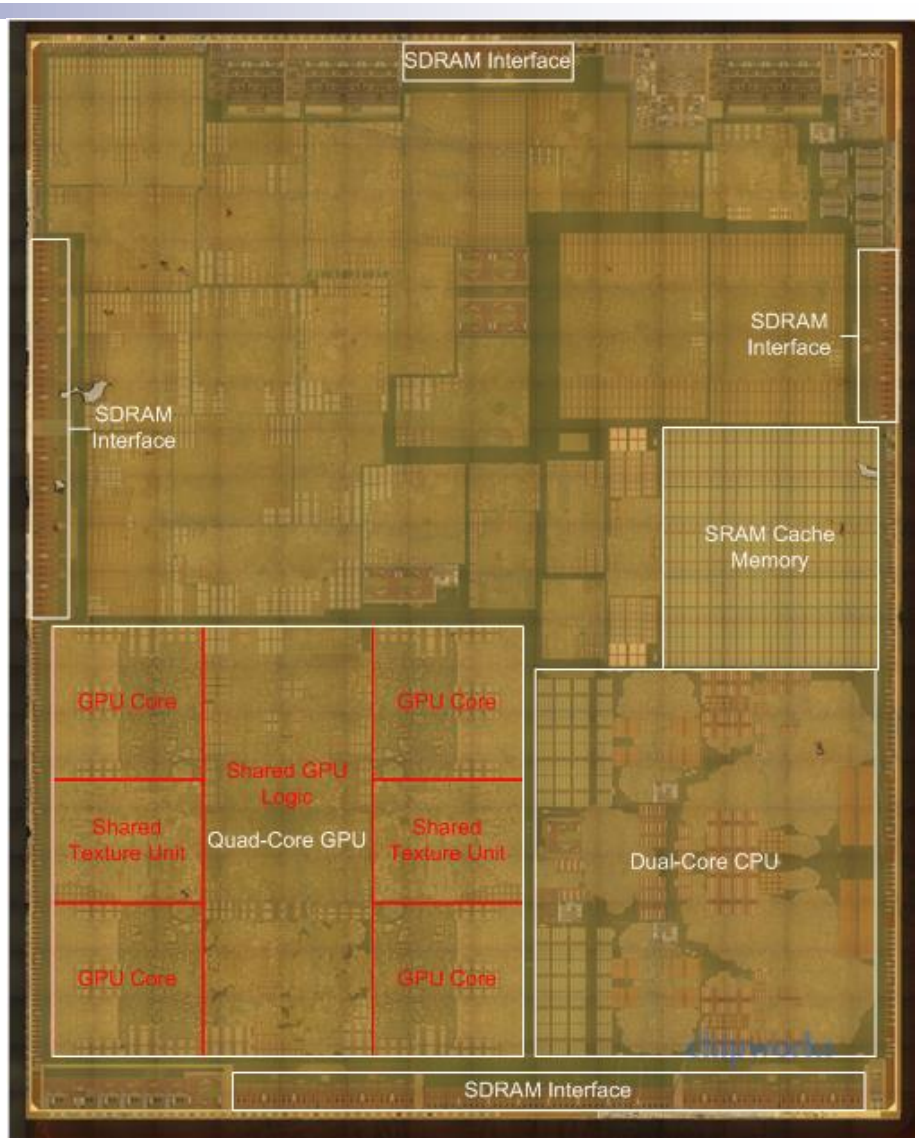
Processor

Package on Package

Main memory (SDRAM)



Inside of the Processor



Inside the Processor (CPU)

- Datapath: performs arithmetic operations on data
- Control: tell datapath, memory, ...what to do
- Memory: where the programs are kept when they are running
- Cache memory
  - Small fast SRAM memory for immediate access to data

What is the Performance?

Plane	DC to Paris	Speed	Passengers	Throughput (pas*mph)
Boeing 747	6.5 hours	610 mph	470	286,700
Concorde	3 hours	1350 mph	132	178,200

- *Which of the planes has better performance*
  - The plane with the higher speed is **Concorde**
  - The plane with the larger capacity is **Boeing 747**

Performance Example

- Time of Concorde vs. Boeing 747
  - Concord is 1350 mph / 610 mph = 2.2 times faster
- Throughput of Concorde vs. Boeing 747
  - Boeing is 286,700 pmph / 178,200 pmph = 1.6 times faster
- Boeing is 1.6 times faster in terms of throughput
- Concord is 2.2 times faster in terms of flying time
- When discussing processor performance, we will focus primarily on execution time for a single job - why?

Defining Performance

- Response time (Execution time)- the time between the start and completion of a task
  - How long it takes to do a task
  - How long does it take for my job to run?
  - How long does it take to execute a program?
  - How long must I wait for a database query?
  - This is also referred to as execution time
- Throughput - the number of tasks completed per unit time
  - How many jobs can the machine run at once?
  - What is the average execution rate?
  - How much work is getting done?

Throughput and Response Time

- Example - Do the following changes to a computer system increase throughput, decrease response time, or both?
  - 1. Replacing the processor in a computer with a faster version
  - 2. Adding additional processors to a system that uses multiple processors for separate tasks—for example, searching the web
- Answer:
  - in case 1, both response time and throughput are improved.
  - In case 2, no one task gets work done faster, so only throughput increases.

Response Time vs. Througput

- Take example of hot dog stand
  - It takes 3 steps to make a hot dog
    - 2 min to take order verbally
    - 2 min to grill hot dog sausage
    - 2 min to wrap up, take money and serve
- Questions:
  - What is the response time for a single customer?
    - 6 min
  - What is the throughput per minute?
    - 1 / 6 = ~0.167
  - Can you think of ways to shorten response time and improve throughput?

Relation between Performance and Execution time

- For some program running on machine X
- Define Performance = 1/Execution Time$_X$
- Relative Performance: "Computer X is n time faster than computer Y"

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

- Example:
  Computer A runs a program in 10 seconds
  Computer B runs the same program in 15 seconds
  how much faster is A than B?
- Solution:
  Performance$_A$ / Performance$_B$
  = Execution Time$_B$ / Execution Time$_A$
  = 15s / 10s = 1.5
  So A is 1.5 times faster than B

Measuring Execution Time

- Elapsed time, Execution time
  - The total response time to complete a task
  - time spent executing on the CPU, accessing disk and memory, waiting for I/O and other processes, and operating system overhead.
  - Determines system performance
- CPU execution time:
  - Total time a CPU spends computing on a given task
  - excludes time for I/O or running other programs
  - This is also referred to as simply CPU time.
    - Comprises user CPU time and system CPU time
    - Determine the CPU performance(user CPU time)
- *We will focus on CPU performance on this course*

CPU Time

- Unix `time` command reflects this breakdown by returning the following when prompted:

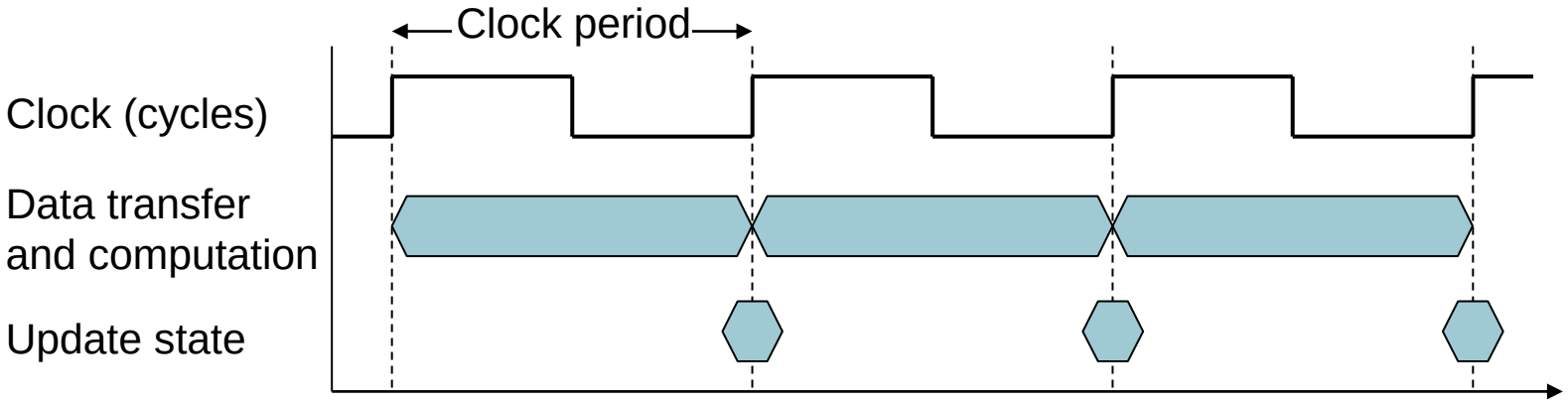
```
90.7s 12.9s 159s 65%
```

Interpretation:

- User CPU time is 90.7 s
- System CPU time is 12.9s
- Elapsed time is 159 s (≠ 90.7+12.9)
- CPU time is 65% of total elapsed time

CPU Clocking

- all computers are constructed using a clock that determines when events take place in the hardware.



  - Clock Cycles: These discrete time intervals
  - Clock Period (Clock Cycle Time) : duration of a clock cycle
  - Clock Frequency (Clock Rate): cycles per second

Clock cycle vs. clock rate?

- Clock period = clock cycle time= cycle length
- Clock rate = clock frequency
- How they are related?

$$1\ \text{sec} = clock\ period * clock\ frequency$$

Units for Clock Period and Frequency

- Matching pairs of units

Clock Period (Clock Cycle Time)	Clock Frequency (Clock Rate)
1 second (1 s)	1 Hertz (1 Hz)
1 millisecond (1 ms) = 10^{-3} second	1 Kilohertz (1 KHz) = 10^{3} Hz
1 microsecond (1us) = 10^{-6} second	1 Megahertz (1 MHz) = 10^{6} Hz
1 nanosecond (1ns) = 10^{-9} second	1 Gigahertz (1 GHz) = 10^{9} Hz
1 picosecond (1 ps) = 10^{-12} second	1 Terahertz (1 THz) = 10^{12} Hz

- Examples
  - Clock Cycle Time: 250ps = 0.25ns = 250×10^{-12}s
  - Clock Rate: 4.0GHz = 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?

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

Instruction Count and CPI

- Instruction : A command that CPU can understand and execute
- Instruction Count: the number of instructions executed by the program
  - Determined by program, ISA and compiler
- CPI: Average cycles per instruction
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

CPI Example

- We have two implementations of the same instruction set architecture
  ps = 10^{-12}second
- Computer A: Clock Cycle Time = 250ps, CPI = 2.0
- Computer B: Clock Cycle Time = 500ps, CPI = 1.2
- Which Computer is faster, and by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \textit{Clock}\text{Cycle Time}_A$$
$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$

A is faster…

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

$$\frac{\textit{Performance}_A}{\textit{Performance}_B} = \frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$

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

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

CPI Example

- Alternative compiled code sequences using instructions in classes A, B, C
- Which code sequence executes the most instructions? Which will be faster? What is the CPI for each sequence?

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

CPI Example

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

2+1+2=5 inst.

4+1+1=6 inst.

- 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

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

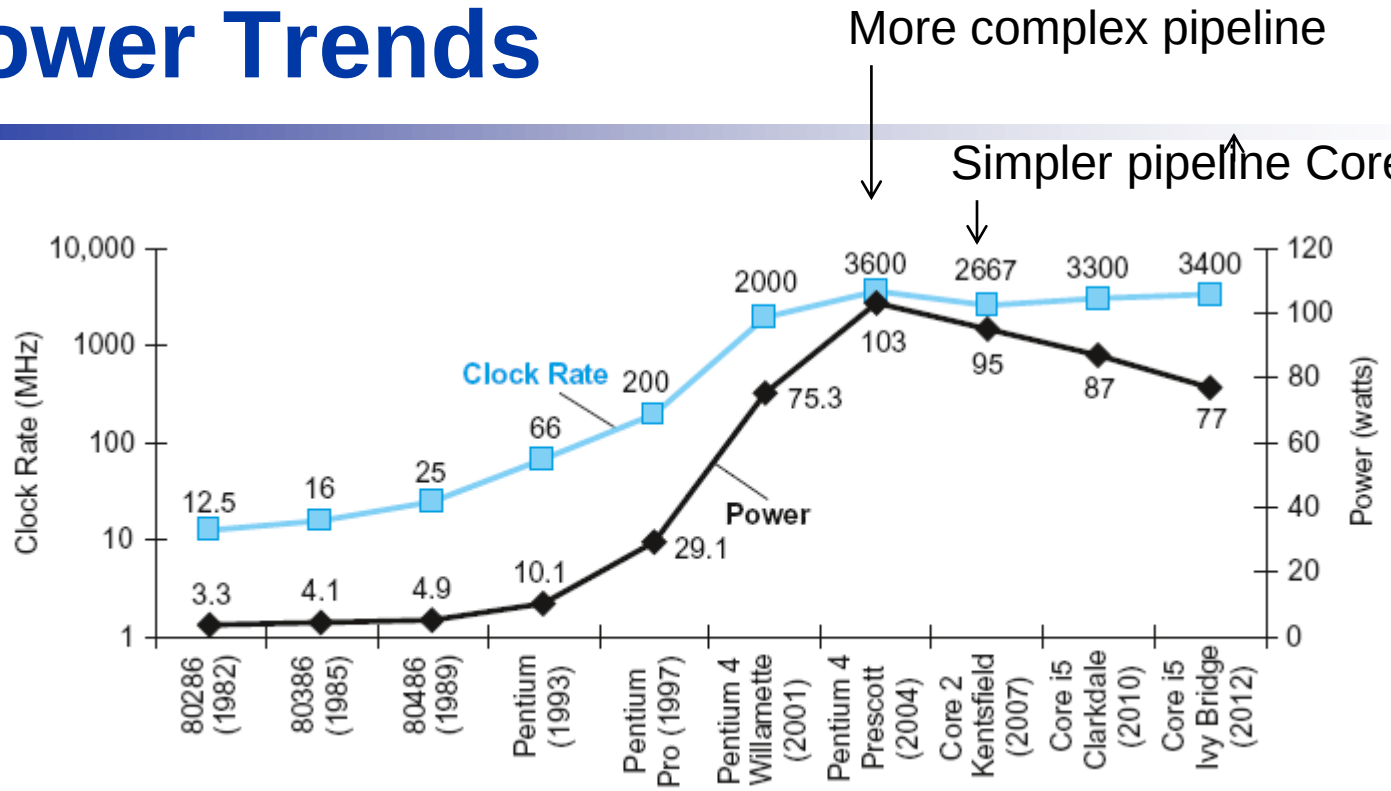
Components of performance	Units of measure
CPU execution time for a program	Seconds for the program
Instruction count	Instructions executed for the program
Clock cycles per instruction (CPI)	Average number of clock cycles per instruction
Clock cycle time	Seconds per clock cycle

The basic components of performance and how each is measured

Performance Summary

- 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

Power Trends



- In CMOS IC technology

CMOS primary energy consumption is dynamic energy, switch on->off; off->on controlled by the clock freq.

$$\text{Power} = 0.5 \times \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

Dynamic Power

×30

5V → 1V

×1000

Reducing Power

- Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^{\ 2} \times F_{old}} = 0.85^4 = 0.52$$

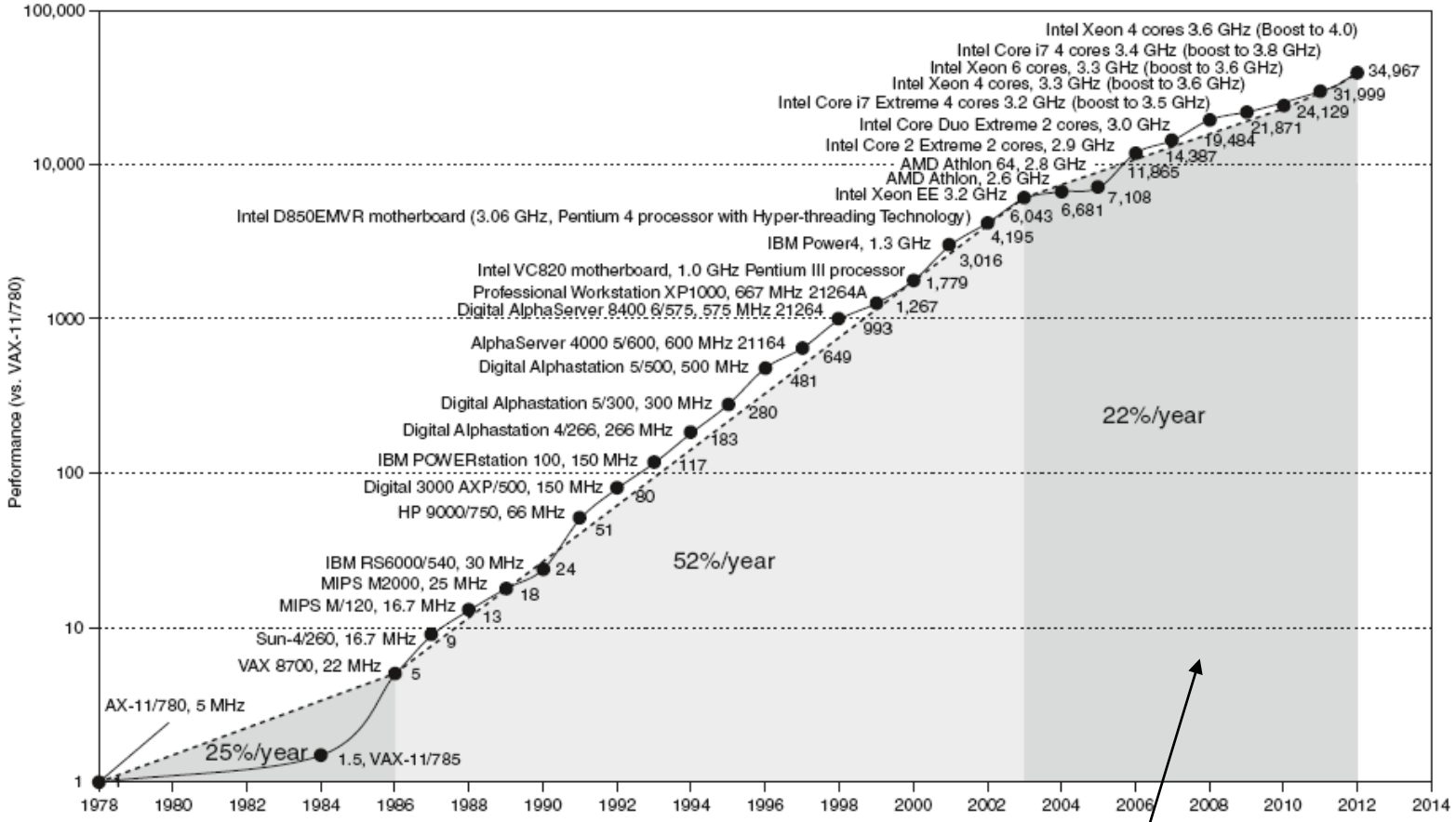
the new processor uses about half the power of the old processor

- The power wall
  - We can't reduce voltage further – make transistor leaky
  - We can't remove more heat
- How else can we improve performance?

Power vs performance

- $Power = Capacitive\ load \times Voltage^2 \times Frequency\ switched$
- Power is a linear function of clock rate
- Performance is a linear function of clock rate
- Power issues: overheating and PMD
- Resolving power issues in past:
  - Decrease voltage → 5v → 1v in 20 years
  - Cooling – remove heat
- The power wall
  - We can't reduce voltage further
  - We can't remove more heat
- How else can we improve performance?
- One possible solution: multi cores
  - Power is a function of $\sqrt{area} = \sqrt{\#cores}$
  - After 2006, desktops and severs are with multi core

Uniprocessor Performance



Constrained by power, instruction-level parallelism, long memory latency

Multiprocessors

- Multicore microprocessors
  - More than one processor per chip
- Requires explicitly parallel programming
  - Compare with instruction level parallelism
    - Hardware executes multiple instructions at once
    - Hidden from the programmer
  - Hard to do
    - Scheduling the sub-tasks
    - Load balancing
    - Optimizing communication and synchronization

Technologies for Building Processors and Memory

- Transistor: an on/off switch controlled by an electric signal
- Integrated Circuit: combining dozens to hundreds of transistors into a single chip.

- How to manufacture integrated circuits?
  - Silicon: called semiconductor
    - Excellent conductors
    - Excellent insulators
    - As a switch

Manufacturing ICs

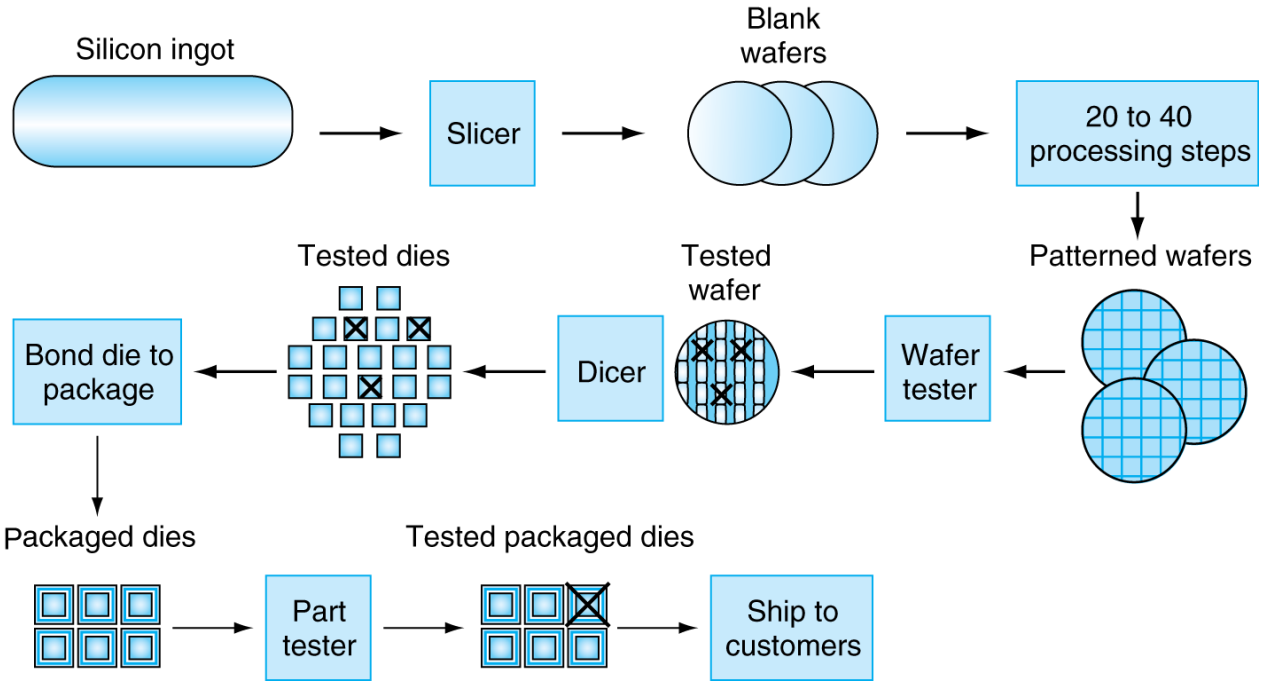


FIGURE 1.12 The chip manufacturing process. After being sliced from the silicon ingot, blank wafers are put through 20 to 40 steps to create patterned wafers (see Figure 1.13). These patterned wafers are then tested with a wafer tester, and a map of the good parts is made. Then, the wafers are diced into dies (see Figure 1.9). In this figure, one wafer produced 20 dies, of which 17 passed testing. (X means the die is bad.) The yield of good dies in this case was 17/20, or 85%. These good dies are then bonded into packages and tested one more time before shipping the packaged parts to customers. One bad packaged part was found in this final test.

- Yield: proportion of working dies per wafer

Integrated Circuit Cost

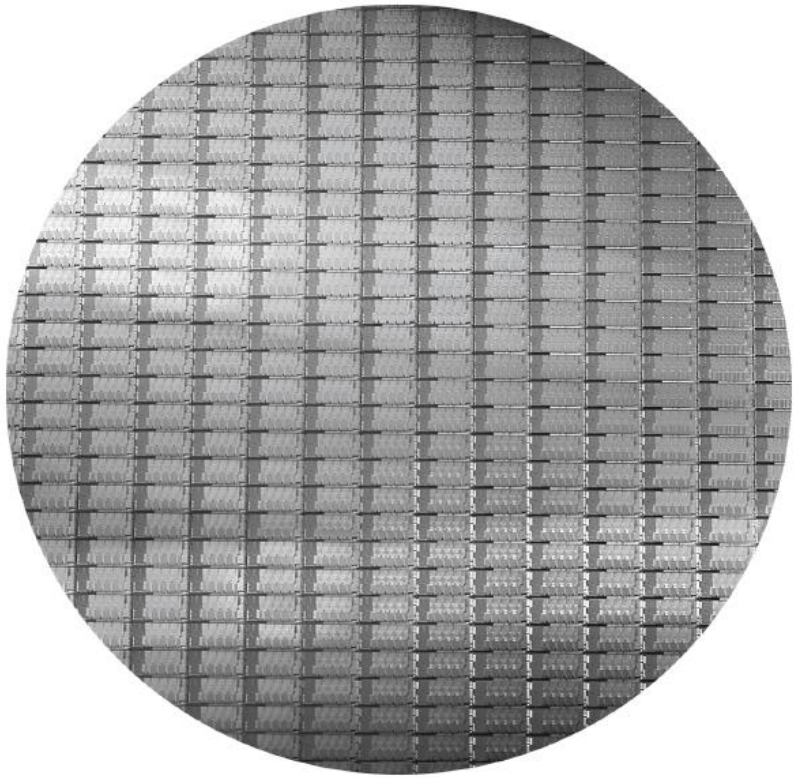
$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$$

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

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

- Nonlinear relation to the area and defects rate
  - Wafer cost and area are fixed
  - Defect rate determined by manufacturing process
  - Die area determined by architecture and circuit design

Intel Core i7 Wafer



- 300mm wafer
  - 32nm technology, 280 chips, 20.7 by10.5 mm
  - 7nm technology, 506 chips, 11.4 by 10.7mm

How to make a IC

Other Performance Factors - Cost

- Initial one-time design cost

$$\text{Design cost per chip} = \frac{Total\ design\ cost}{Total\ shipping\ volume}$$

- Per-chip manufacturing cost
  - Super-linear function of die size, which equal number of transistors divided by process technology

SPEC CPU Benchmark

- Programs used to measure performance
  - Supposedly typical of actual workload
- System Performance Evaluation Cooperative (SPEC)
  - Develops benchmarks for CPU, I/O, Web, …
- SPEC CPU2006
  - Elapsed time to execute a selection of programs
    - Negligible I/O, so focuses on CPU performance
  - Normalize relative to reference machine
  - Summarize as geometric mean of performance ratios
    - CINT2006 (integer) and CFP2006 (floating-point)

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

CINT2006 for Intel Core i7 920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

Pitfall: Amdahl's Law

- Improving an aspect of a computer and expecting a proportional improvement in overall performance

$$T_{\text{improved}} = \frac{T_{\text{affected}}}{\text{improvement factor}} + T_{\text{unaffected}}$$

- Example: multiply accounts for 80s/100s
  - How much improvement in multiply performance to get 5× overall?

$$20 = \frac{80}{n} + 20$$

  - Can't be done!
- Corollary: make the common case fast

Pitfall: MIPS as a Performance Metric

- MIPS: Millions of Instructions Per Second
  - Doesn't account for
    - Differences in ISAs between computers
    - Differences in complexity between instructions
    - Vary between programs

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6}$$

$$= \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

Concluding Remarks

- Hierarchical layers of abstraction
  - In both hardware and software
- Instruction set architecture
  - The hardware/software interface
- Execution time: the best performance measure
- Power is a limiting factor
  - Use parallelism to improve performance

Number System

Table 2.1 Example of five number systems.

System	Radix (base)	Digits
Binary	2	0, 1
Ternary	3	0, 1, 2
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Decimal Numbers

- Base is 10
- we need 10 different digits for each position (0 – 9)
- Zero is always the first number.
- When we count, we usually count "00, 01, 02, 03, 04, 05 , 06, 07, 08, 09, **1**0, **1**1, **1**2, ...“
- Each digit to the left and right of the decimal point is given a name which identifies that digit's placeholder.
- Each placeholder is a multiple of ten.
- For now lets just consider positive numbers

Decimal Numbers

Each placeholder is a base of ten.

- 10^0 = ones
  - Any number to the zero power is always equal to 1.
  - $n^0=1$
  - $10^0=1$
- 10^1 = tens
  - Any number to the first power is always equal itself.
  - $n^1=n$
  - $10^1=10$
- 10^2 = hundreds
- 10^3 = thousands

THOUSANDS	HUNDREDS	TENS	ONES
7	4	0	8

Decimal Numbers

Number	7	4	0	8
Position Name	Thousands	Hundreds	Tens	Ones
Exponential Expression	10^3*7	10^2*4	10^1*0	10^o*8
Calculated Exponent	1000*7	100*4	10*0	1*8

Sum of the powers of ten.

7 *1000 + 4*100 + 0 * 10 + 8 * 1 = 7408

Decimal Numbers

- What is value of 31415?
- Precisely it is really

$3 * 10^4$	30,000
$1 * 10^3$	1,000
$4 * 10^2$	400
$1 * 10^1$	10
$5 * 10^0$	5

The expansion form of 31415 is

$$31415 = 3 * 10^4 + 1 * 10^3 + 4 * 10^2 + 1 * 10^1 + 5 * 10^0$$

Binary Number

- For binary numbers, base is 2
- We only have 2 digits for each position, 0 or 1
- Binary system is based on multiples of two.
- In binary numbering the numbering scheme repeats after the second digit.
- Let's count to five in binary: “0000, 0001, 0010, 0011, 0100, 0101“
- Binary numbering includes names for digit placeholders

Conversion from Binary to Decimal

- 1101_2 to ?

Number	1	1	0	1
Position Name	Eights	Fours	Twos	Ones
Exponential Expression	2^3*1	2^2*1	2^1*0	2^0*1
Calculated Exponent	8*1	4*1	2*0	1*1

- Sum of the powers of two
- $1101_2 = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 13$

Conversion from Binary to Decimal

Number	1	1	0	1
ON/OFF	On	On	Off	ON
Exponential Expression	2^3*1	2^2*1	0	2^0*1
Calculated Exponent	8*1	4*1	0	1*1
Solved Multiplication	8	4	0	1
Add to calculate Value	8+4+0+1=13			

Conversion from Decimal to Binary

- 97?

Decimal Number=97			
Division Expression	Quotient	Remainder	Direction
97/2	48	1	
48/2	24	0	
24/2	12	0	
12/2	6	0	
6/2	3	0	
3/2	1	1	
1/2	0	1	
Binary Number=1100001			

Hexadecimal Numbers

- It is much easier to work with large numbers using hexadecimal values than decimal or binary.
  - One Hexadecimal digit = 4bits
  - Two hexadecimal digits = 8 bits
  - This makes conversions between hexadecimal and binary very easy
- For hexadecimal numbers, base is 16
  - We need 16 digits to represent a single position
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- To differentiate from decimal numbers
  - We can put 16 subscript after the number, e.g., $A13CD902_{16}$

Conversion from Hexadecimal to Decimal

- 11A8 to ?

1	1	A=10	8
16^3	16^2	16^1	16^0
4096*1	256*1	16*10	8*1
4096	256	160	8
4096+256+160+8= 4520			

This example shows the conversion for R > 10.

$$(15C.F)_{16} = (1 \times 10^2 + 5 \times 10^1 + C \times 10^0 + F \times 10^{-1})_{16}$$

$$= (1 \times 16^2 + 5 \times 16^1 + 12 \times 16^0 + 15 \times 16^{-1})_{10} = (348.9375)_{10}$$

Convert Hexadecimal to Binary

- Convert *each* hexadecimal digit into its *4-bit* binary equivalent
- Example: 1AB

Hex	1	A	B
Bin	0001	1010	1011
000110101011			

Convert Binary to Hexadecimal

- Convert *each* 4bit binary digit into its *hexadecimal* equivalent
- starting from the right
- If there is an odd number of bits, add zeros to the left to make a complete 4bit digit.

- Example:110101011

Bin	0001	1010	1011
Hex	1	A	B
1AB			

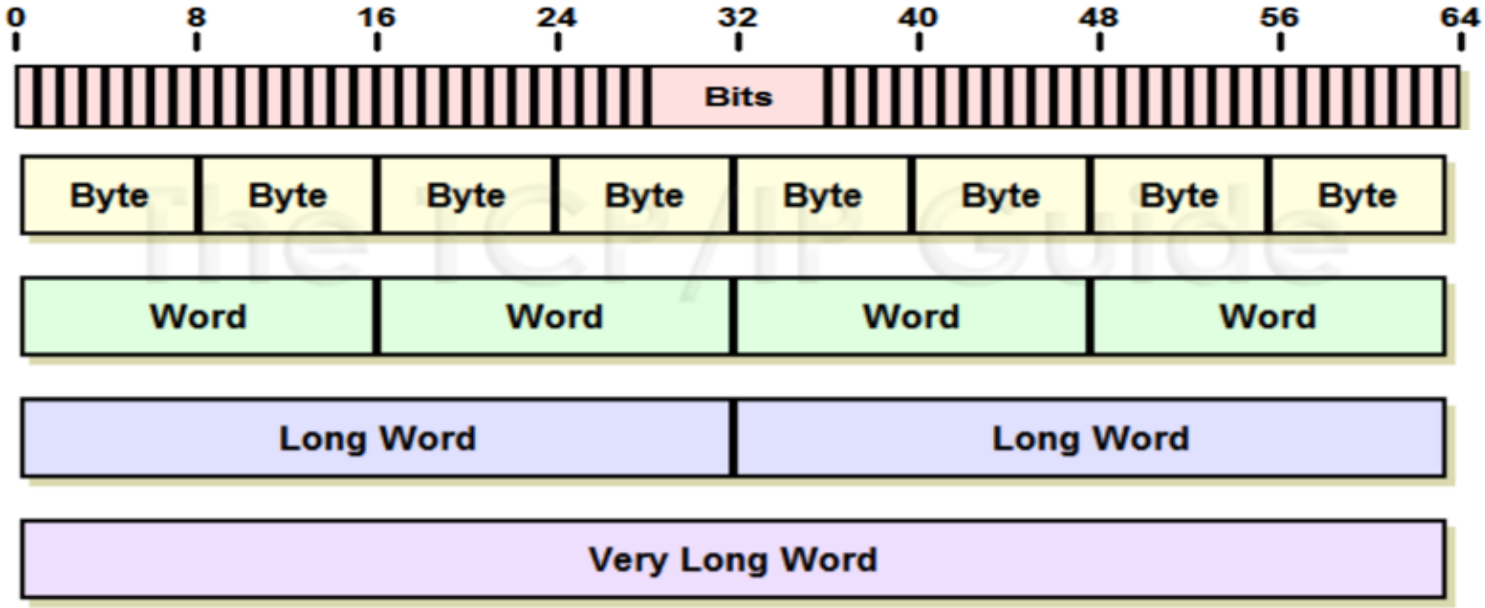
Binary Representations

- Important Fact 1: If you are going to represent any type in k bits, you can only represent 2^k different values.

- Important Fact 2: The same bit string can represent an integer (signed or unsigned), float, character string, list of instructions, address, etc. depending on the context.

Bits Aren't So Convenient

- A bit is single digit in binary number, 0 or 1
- we group them together into larger units.
- Sizes of these units depends on the *architecture / language.*



An Example

- Computer storage is generally measured and manipulated in bytes and collections of bytes.
  - A kilobyte, or KB, is 1,024 bytes
  - A megabyte, or MB, is 1,0242 bytes
  - A gigabyte, or GB, is 1,0243 bytes
  - A terabyte, or TB, is 1,0244 bytes
  - A petabyte, or PB, is 1,0245 bytes
- Computer manufacturers often round off these numbers
  - a megabyte is 1 million bytes and a gigabyte is 1 billion bytes