IT Compact Course

Hardware and Software

Internet and Web

Cryptography

Kaspar Etter, October 2011

1. Hardware and Software

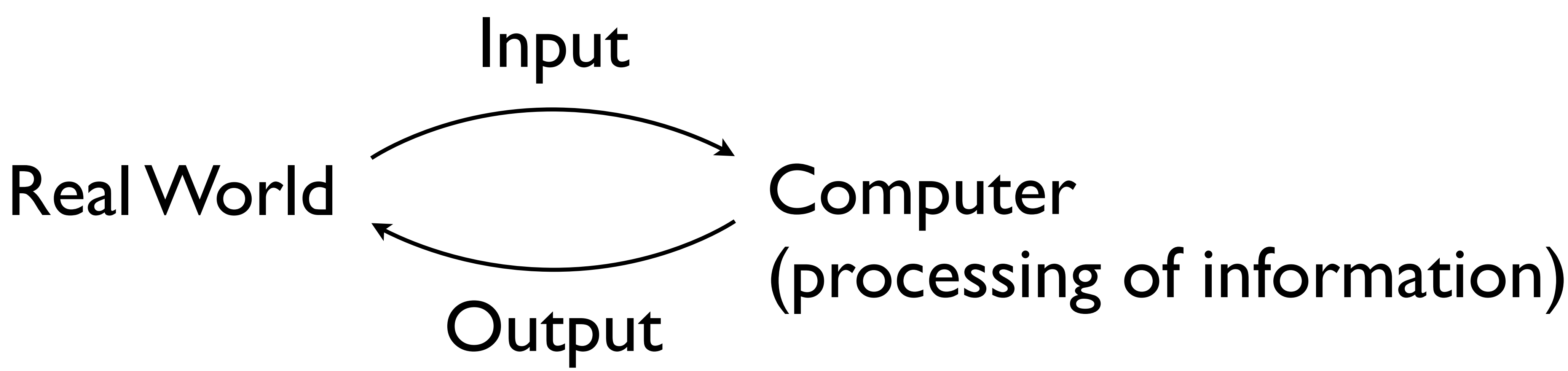
Hardware:

- Computers (physical)
- General purpose machines
- Expensive to design
- Expensive to copy
- Subjected to wear

Software:

- Programs (intangible)
- Special purpose instructions
- Expensive to design
- Free to copy
- Wear-free

Input/Output (I/O)



«Computer Science is no more about computers than astronomy is about telescopes.»

Edsger Dijkstra (misattributed)

Computers

The good news are that …

- they do exactly what you tell them to do
- they do it very fast

The bad news are that …

- they do exactly what you tell them to do
- they do it very fast

«To err is human, but to really mess things up you need a computer!»

Outline

1.1. Processor
1.2. Memory
} Hardware

1.3. Program
1.4. Operating System
} Software

1.5. Data Structures
1.6. Algorithms
} Programming

General reference and source for further reading: www.wikipedia.org

1.1. Processor

- Central Processing Unit (CPU)
- Sequential processing of arithmetic and logical operations
- Data stored as binary numbers due to easy implementation in digital electronic circuitry using logic gates
- Only integers considered here (no floating-point numbers)
- A *digital* system uses *discrete* values, an *analog* system uses *continuous* values to represent information (*Digital* comes from the Latin word *digitus*, meaning finger)
- A *bit* (a contraction of *binary digit*) is the basic unit of information in computing and is usually denoted as 0 and 1

Bits and Bytes

- 8 bits (b) = 1 byte/octet (B), allows to represent 256 values
- Unsigned 8-bit integer: 0 to 255; signed integer: -128 to 127
- Byte was the # of bits to encode a single character of text: Basic addressable element in many computer architectures
- Processors manipulate bits in fix-sized groups named *words*
- Prefixes:

Decimal (SI)		Binary	
kilo (k)	10^3	kibi (Ki)	$2^{10} \approx 1.02 \cdot 10^3$
mega (M)	10^6	mebi (Mi)	$2^{20} \approx 1.05 \cdot 10^6$
giga (G)	10^9	gibi (Gi)	$2^{30} \approx 1.07 \cdot 10^9$
tera (T)	10^{12}	tebi (Ti)	$2^{40} \approx 1.10 \cdot 10^{12}$

Pointers, Registers and Flags

- Von Neumann architecture: Data & code in same memory
- Material based on Intel's instruction set architecture *x86-32*
  Heavily simplified (no segmentation, addressing modes, etc.)
- Word length of 32 bits (= 4 bytes), from 0 to 4'294'967'295
- Instruction pointer: Memory address of next instruction
- 8 registers hold the current operands (the first 4 being general-purpose): EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP
- Carry, overflow and zero flag: Bits set after every operation
- Much of the trouble comes from backward compatibility

Operations

- Conceptually, there are only three types of operations:
  - Load data from memory to registers and store data from registers back to memory
  - Perform arithmetic and logical operations on registers
  - Control program flow with (conditional) jumps in code
- Memory is accessed with pointers to the desired locations
- Jumps can be absolute or relative in terms of memory address and often depend on the last executed operation
- The x86 instruction set comprises hundreds of operations

Assembly

- A low-level programming language that represents binary machine code in a human-readable form (with mnemonics)
- Needs to be translated into machine code for execution
- Example: Add together all numbers from 1 to 100

operator — operands — comment

```
      mov #0, sum     ; set sum to 0
      mov #1, num     ; set num to 1
loop: add num, sum    ; add num to sum
      add #1, num     ; add 1 to num
      cmp num, #100   ; compare num to 100
      ble loop        ; if num <= 100, go back to 'loop'
      halt            ; end of program. stop running
```

label — branch less or equal — source and destination (register)

Pipelining

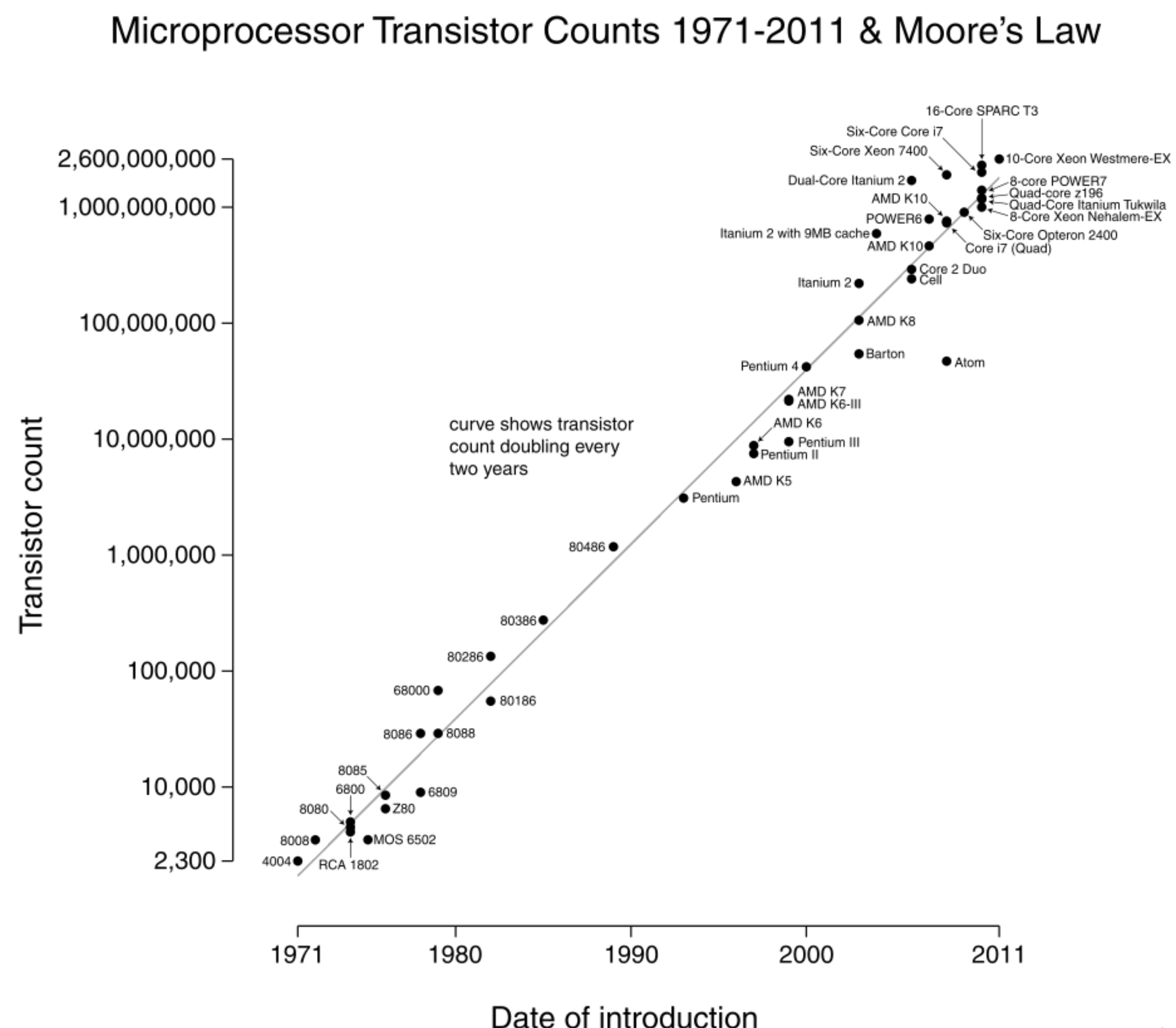
- Increase the instruction throughput by splitting the processing of an instruction into a series of independent steps (which increases the time to execute a single instruction)
- Issue instructions at the processing rate of the slowest step
- Maintain semantics for interdependent instructions and branches (branch prediction and speculative execution)

Instr No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	**1**	**2**	**3**	**4**	**5**	**6**	**7**

1. Instruction fetch
2. Instruction decode and register fetch
3. Execute
4. Memory access
5. Register write back

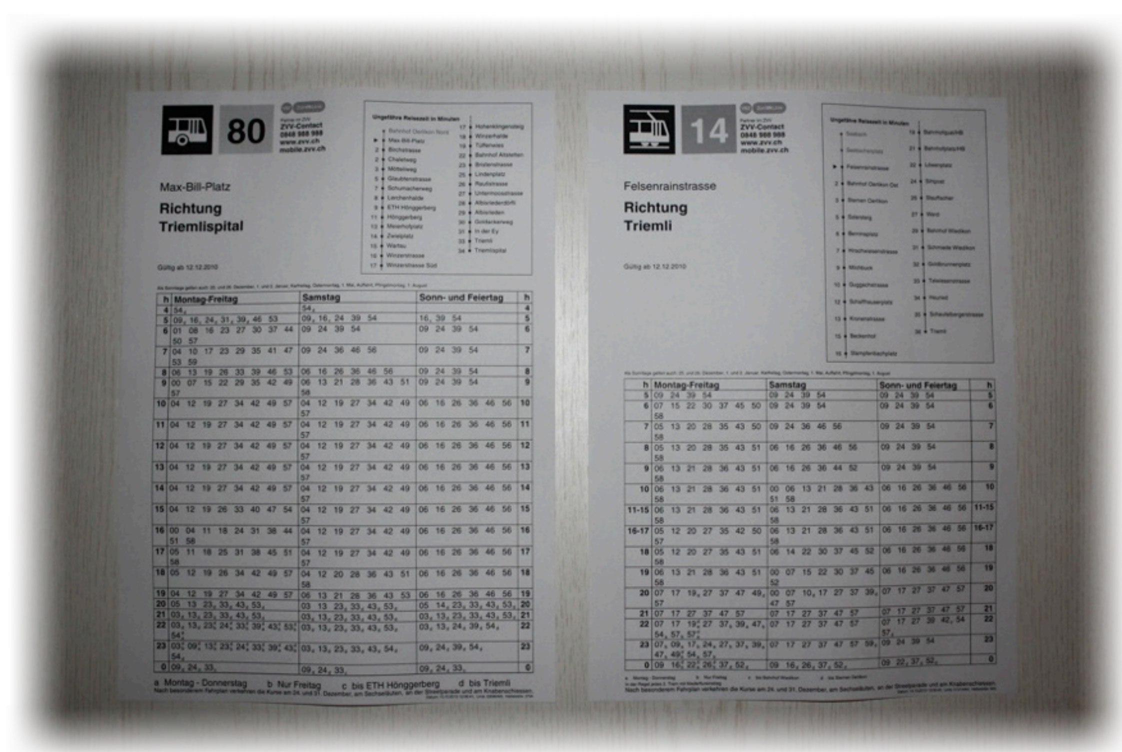
Moore’s Law

- Trend described by Intel co-founder Gordon Moore, 1965: The number of transistors on chips doubles every 2 years.
- Originally an observation and forecast, now a self-fulfilling prophecy
- Wirth’s law, 1995: Software is getting slower more rapidly than hardware becomes faster.



1.2. Memory

- A list of cells into which numbers can be placed or read
- The cells are numbered and can be addressed accordingly
- Hardware does not know the semantics of these numbers
- Memory is the bottleneck (limiting component of a system)



Schedule of bus and tram departures

Caching

- A cache stores data for faster access in the future
- Cache hit: Requested data is contained in the cache
- Cache miss: Data has to be fetched from another location; Replaces a cache entry selected by the replacement policy
- The writing of data can be handled in two ways:
  - Write-through: Write to cache and memory concurrently
  - Write-back: Store the dirty cache entry on replacement

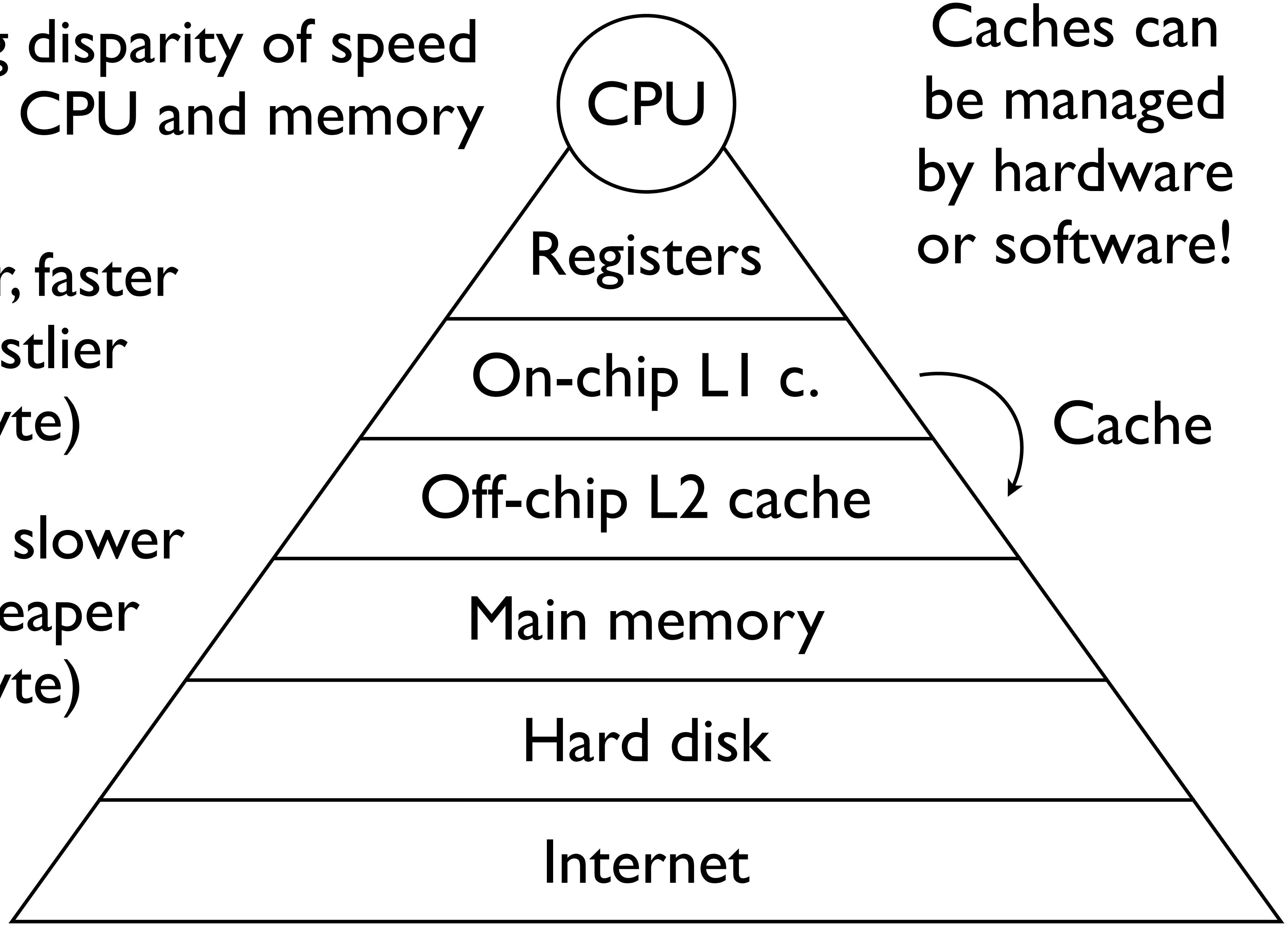
  Entries become stale, if somebody updates the original data
- Useful due to temporal and spatial locality of references
- The hit ratio (#hits / #misses) determines the performance

Memory Hierarchy

Growing disparity of speed between CPU and memory

Smaller, faster and costlier (per byte)

Larger, slower and cheaper (per byte)



Caches can be managed by hardware or software!

1.2. Memory

Characteristics and Access

- ROM: Read-only memory, e.g. Compact Discs (CD-ROM); Used for firmware (low-level, hardware-specific software)
- RAM: Random-access memory, data can be accessed in any order (unlike disks); Often volatile storage (power supply!)
- Main memory connected to the CPU via a memory bus
- The memory management unit (MMU) calculates the actual memory address
- Disks: Seek time + rotation

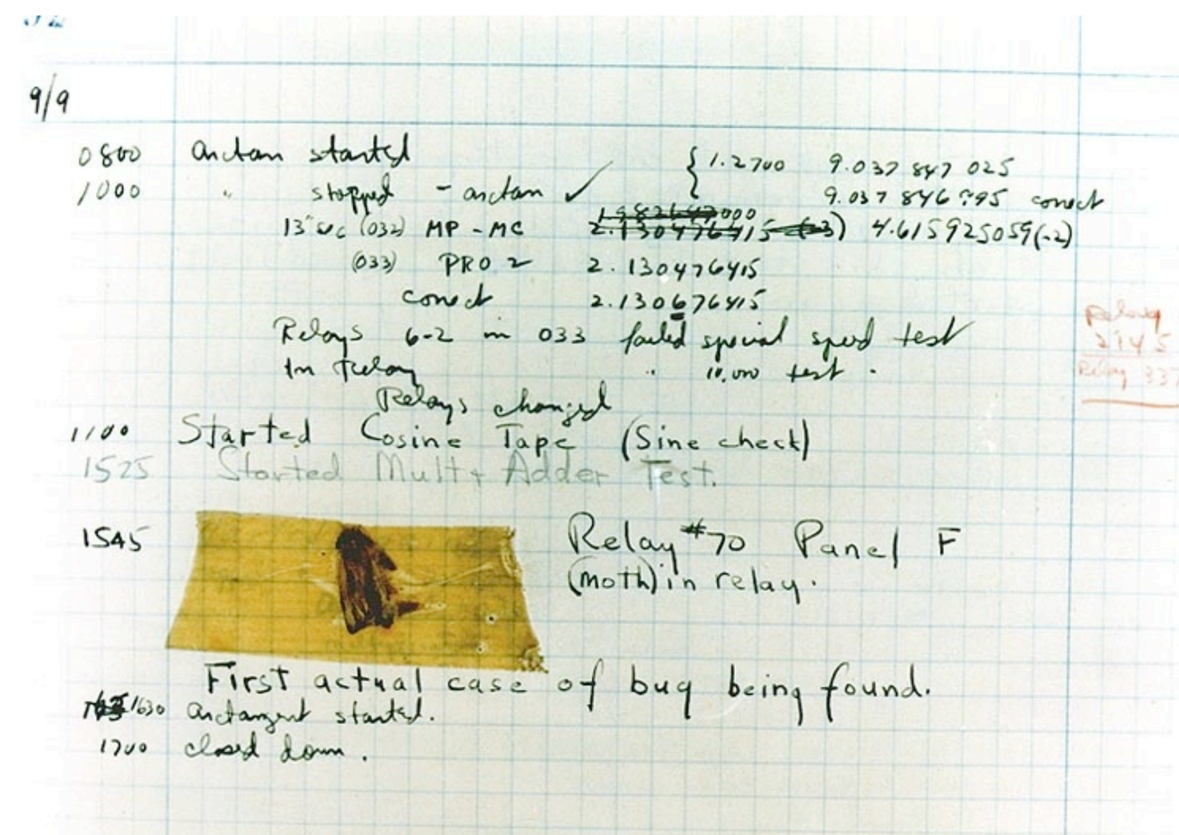
1.3. Program

- A sequence of instructions that perform a specified task
- In its simplest form, takes some input and generates output
- The main qualities of software (programs) are:
  - Correctness: functional behavior according to specification
  - Performance: fast execution, low memory consumption
  - Maintainability: easy modification after initial development
  - Reusability: simple adaptation to new purposes/products
  - Usability: learnability, efficiency, memorability, satisfaction
  - Security: confidentiality, integrity, availability (cryptography)

Bugs

«Any feature is a bug unless it can be turned off.»

- Software errors cost the US economy $60 billion annually, or about 0.6 percent of the gross domestic product (2002)
- US$1 billion Ariane 5 rocket destroyed after takeoff (1996)
- Several patients were killed by the Therac-25 radiation therapy machine (1980s)
- Terminology: debug, buggy
- «Testing shows the presence, not the absence of bugs.» Edsger Dijkstra



«First actual case of bug being found.» (1947)

Programming Language

- An artificial language to give instructions to a computer
- Developing software in assembly language is error-prone
- Increase the programmer's productivity by providing a tool: A program that mediates between man and machine
- Natural language is inappropriate: complex and ambiguous
- Formal languages consist of two parts: syntax and semantics
- Defined by a specification or a reference implementation
- Trade-off between abstraction/safety and expressiveness
- Benefit: High-level languages reduce platform dependency

1.3. Program

Source Code

- Text written in a high-level programming language that can be translated to binary machine code for execution
- Protected by copyright, protected form of free speech
- Example: "Hello world" in the C programming language

```
#include <stdio.h>

int main() {
  printf("Hello world!\n");
  return 0;
}
```

include standard I/O library

print formatted string

main method with return type integer (called by the run-time environment)

terminates statement

code block

escape sequence for newline character

exit code indicating successful execution

Compiler and Interpreter

- A compiler translates source code into machine code
- It checks the syntax and rejects invalid programs
- Semantic checks performed either statically (at compile time, e.g. types) or dynamically (at run time, e.g. arrays)
- Optimization: Constant propagation, common subexpression elimination, register allocation, instruction scheduling
- Interpreter: Execute code directly, present during execution
- Just-in-time compilation (JIT): Sections compiled 'on the fly'
- Reverse engineering: Decompilation (vs. code obfuscation)

Text Terminal

- Command-line interface (CLI): Type commands to interact
- Graphical user interface (GUI): Manipulate visual elements
- Commands invoke programs with standard in- and output
- Interrupt their execution with ctrl-c (or ctrl-d on input)
- Structure: *command arguments* (syntax given by command)
- '>' redirects output to a new file ('>>' appends to a file)
- '|' chains output of left command & input of right command
- Prompt: Ready to accept commands, usually ends with '$'
- Mac: Open 'Applications/Utilities/Terminal' and type 'help'

Demo: Integer Factorization

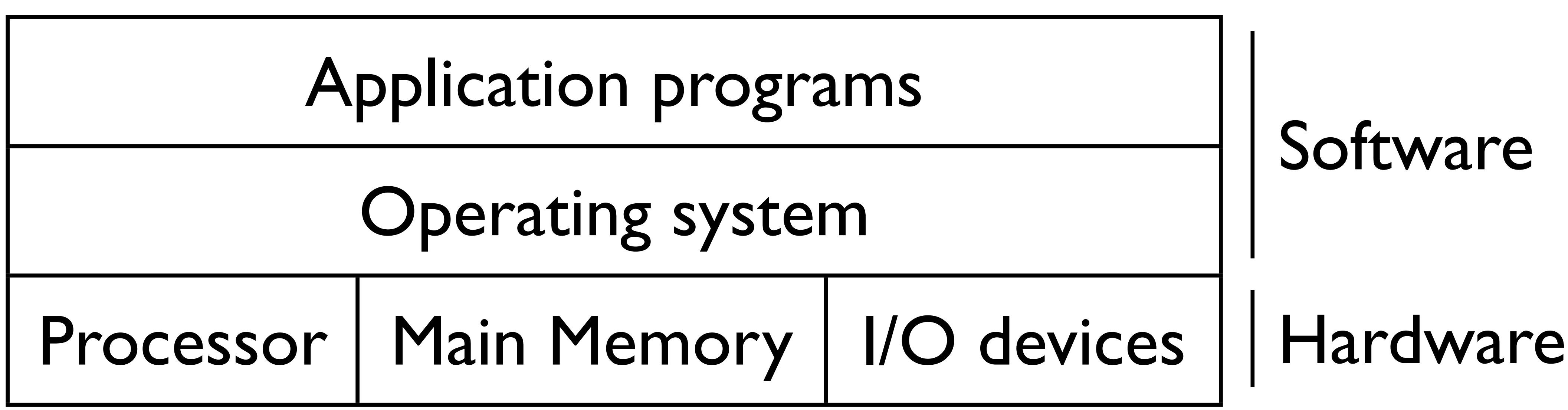
```
#include <stdio.h>                           // Comments:
int main() {
   int number;                               // Variable declaration
   printf("Number: ");                       // Print to standard output
   scanf("%d", &number);                     // Read a number from input
   while (number > 0) {                      // Loop while condition true
      printf("Factors: ");
      int factor = 2;                        // Assign value to variable
      while (factor * factor <= number) {
         if (number % factor == 0) {         // Check remainder (modulo)
            printf("%d, ", factor);
            number = number / factor;
         } else {
            factor = factor + 1;
         }
      }
      printf("%d\n", number);
      printf("Number: ");
      scanf("%d", &number);
   }
   return 0;
}
```

Store in a file ‘code.c’
Compile with ‘gcc code.c’
Run by typing ‘./a.out’

Note: If you want to learn a language, learn Java and not C/C++!

1.4. Operating System

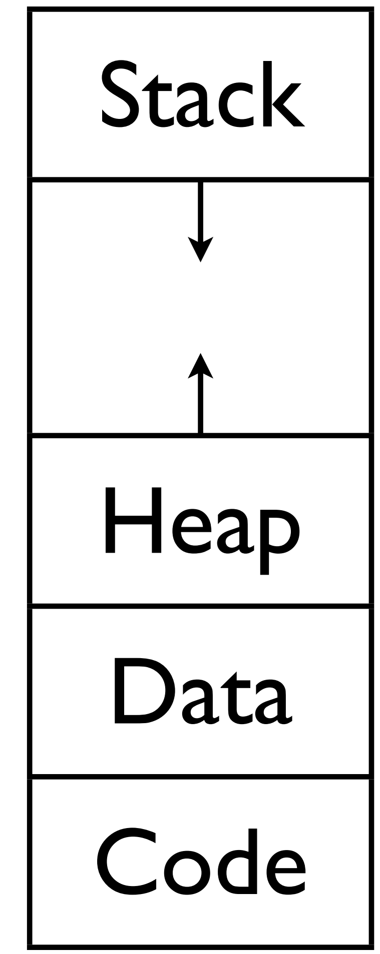
- The OS manages the hardware:
  - Abstraction: Simplify and standardize access to hardware (with so-called Application Programming Interface (API))
  - Duplication: Provide same resources to several programs
  - Protection: Ensure fairness and prevent misbehavior



Process

- The OS creates for every running program a new process
- OS gives to each process the illusion of exclusive hardware: Execution without interruption, own memory and own I/O
- Processes can run concurrently by an interleaved execution
- Control transfer between processes with context switches
- Interrupts trigger special code in OS (privileged execution)
- Process scheduling done with regular hardware interrupts
- Processes have an owner and corresponding permissions
- A process can again have multiple execution units: Threads

Virtual Memory



- Each process has its own virtual address space:
  - Code: Instructions copied from the executable
  - Data: Global variables (statically allocated)
  - Heap: Objects (dynamically allocated memory)
  - Stack: Tracks function calls and local variables
- Virtual memory split into blocks called pages
- OS allocates memory on demand at any location
- Every memory access gets translated into physical address
- If main memory is full, OS swaps inactive pages to hard disk

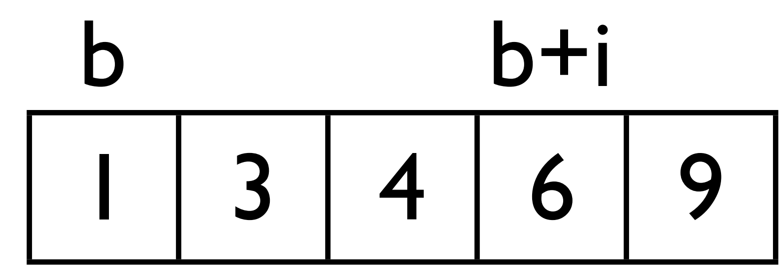
Partitioning

- Divide the hard disk drive into multiple logical storage units
- Booting is the process of loading the OS when starting up
- Multiple OSs can be installed on different partitions, loaded by the built-in basic input/output system (BIOS, "firmware")
- File systems organize data to be retained after termination of a process: Store data permanently in units known as files
- Files consist of linked blocks, are structured by directories
- Specific FS layouts per partition, recover from corrupted FS
- Defragmentation reorganizes files into contiguous blocks

1.5. Data Structures

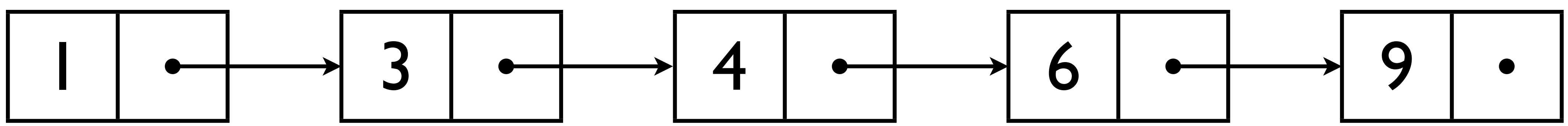
- A data structure is a way of storing and organizing data
- Different data structures suited to different applications
- Goal is time and space efficient manipulation of stored data
- Support the design of efficient algorithms (belong together)

- Arrays are a collection of elements in a continuous block
- Address of each element can be computed from its index
- Size fixed at allocation, insertion or removal needs copying
- Constant access, but linear insertion

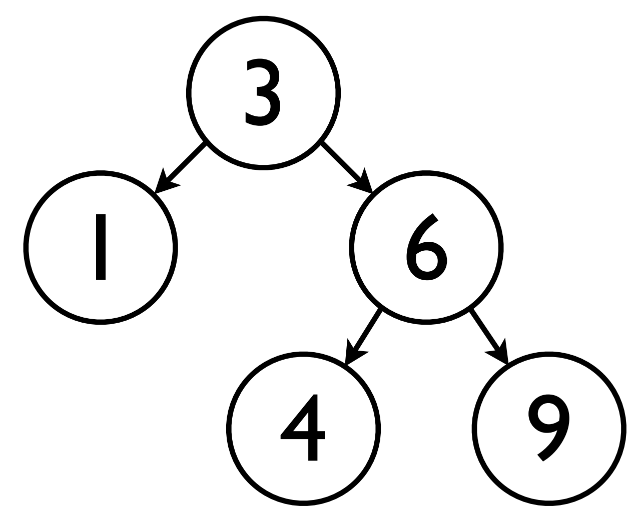


Lists and Trees

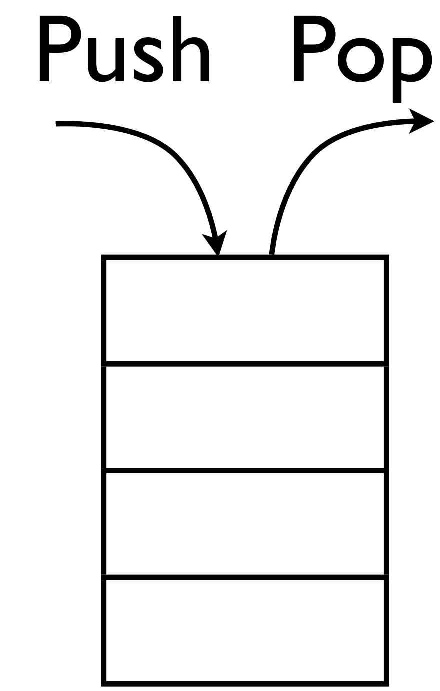
- A linked list is a sequence of elements stored in nodes
- Each node references the next node in the sequence
- Constant insertion and removal, but linear access time



- A binary tree is a linked structure where every node has at most two child nodes
- Restriction that left nodes are smaller and right nodes are bigger allows binary search

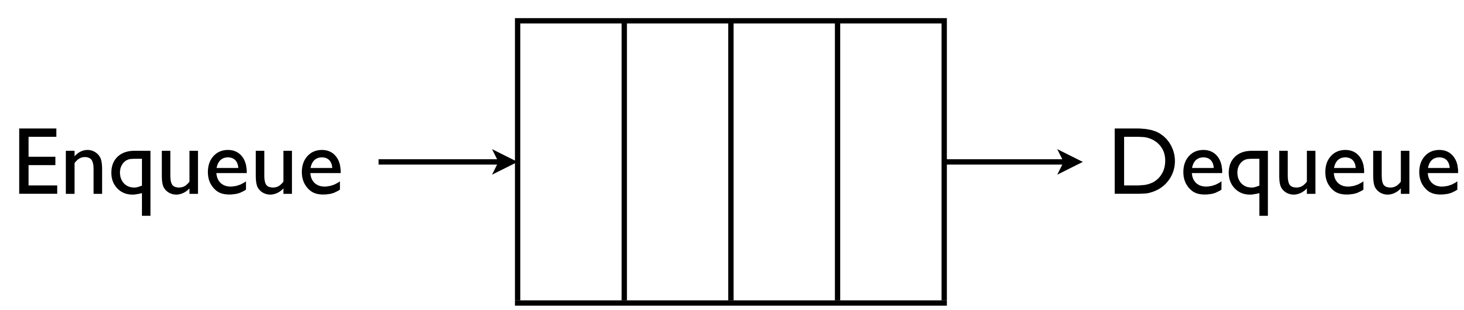


Stacks and Queues



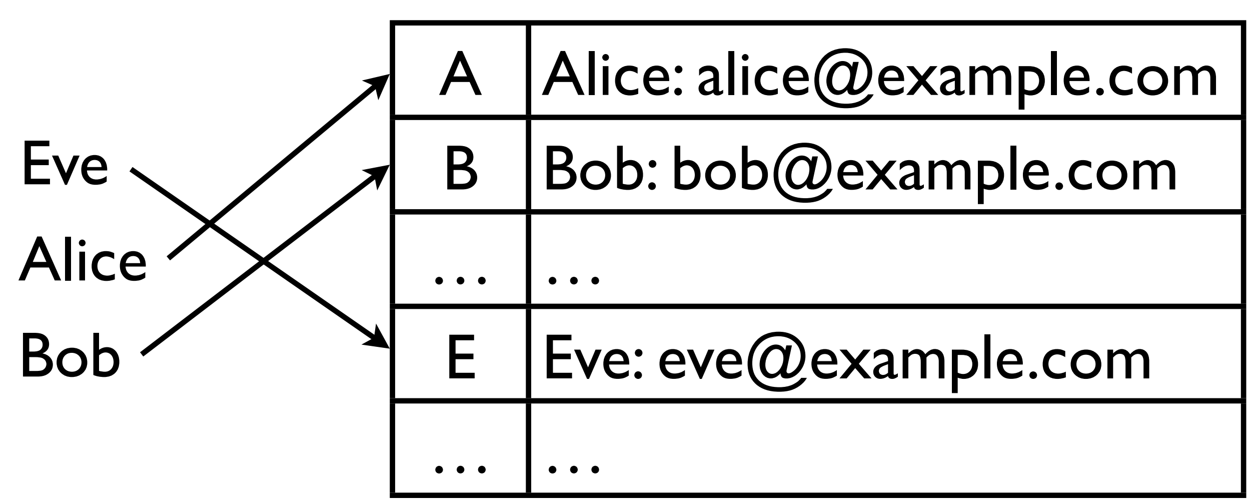
- A stack is a last in, first out (LIFO) collection
- Only two operations provided: Push and pop
- Implementation with an array or a linked list
- Used to track open tasks (e.g. the call stack)

- A queue is a first in, first out (FIFO) collection (as a buffer)
- Only two operations provided: Enqueue and dequeue



Hash Tables

- A hash table maps identifying values to associated values
- A hash function is used to transform the key into an index that indicates the corresponding value's position in an array
- A hash function maps values from a large to a small domain
- Collisions occur when different keys map to the same hash
- Widely used due to near-constant lookups (exc. collisions)



Buffer Overflow

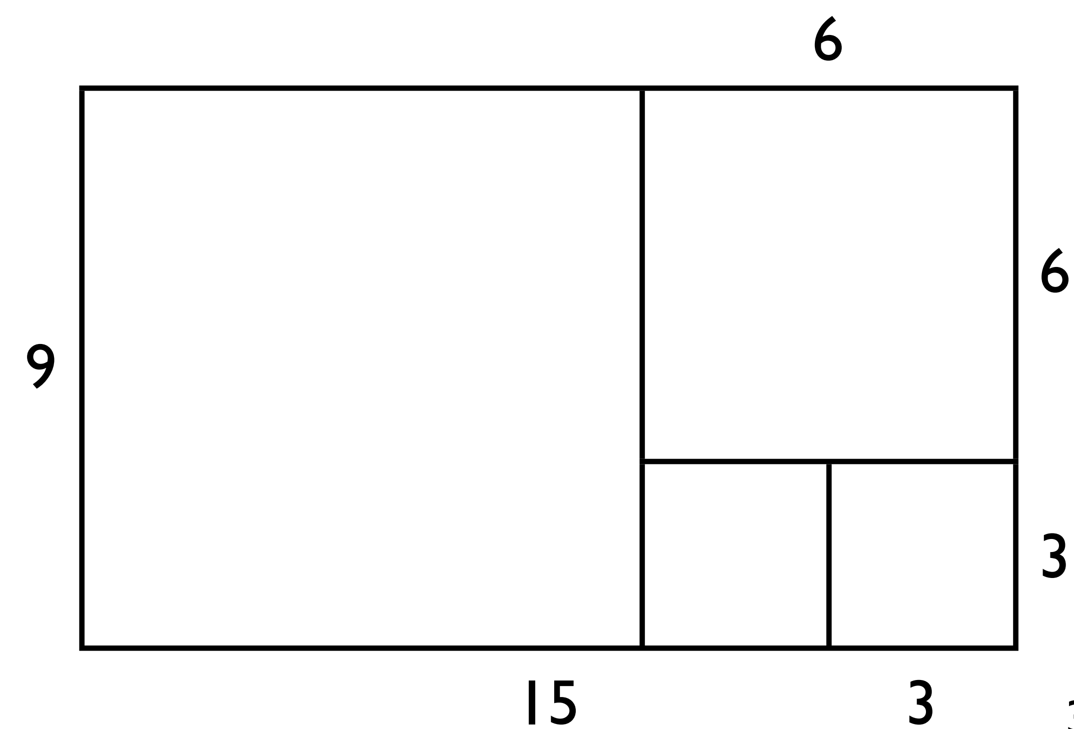
- A buffer temporarily holds data that is moved from one place to another (implemented in hardware or software)
- Typically used for streaming (I/O) with variable rates
- Often implemented as a queue for simultaneous access
- Buffer overflows if incoming data exceeds storage capacity
- If not properly handled, adjacent memory gets overwritten
- Attacker overwrites return address or variable in call stack
- After decades of exploitation, still one of top vulnerabilities
- Solution: Use memory safe programming languages like Java

1.6. Algorithms

- An algorithm is a procedure for solving a specified problem in a finite number of steps (i.e. eventually producing output)
- Transitions between states do not have to be deterministic
- Brute-force: Naïve method of trying every possible solution
- Euclid's algorithm for the greatest common divisor (GCD):

```
function gcd(a, b)
  if a = 0
    return b
  while b ≠ 0
    if a > b
      a := a - b
    else
      b := b - a
  return a
```

a	**b**
15	9
6	9
6	3
3	3
3	0



Sorting

- Given a list of comparable objects, return them in order
- Bubble sort: Steps repeatedly through the list, compares adjacent items and swaps them if they are in wrong order
  - Visualizing the sorting, small elements bubble to the top
  - In-place algorithm: Only constant amount of extra storage
- Merge sort: Divides the list into halves, sorts them recursively and merges the results (divide & conquer algorithm)
  - Preserves the input order of equal elements (stable sort)
  - Requires a linear amount of additional storage space
- See: cs.usfca.edu/~galles/visualization/ComparisonSort.html

Time Complexity

- Estimating the processing time of algorithms, we are only interested in how they respond to changes in input size: Efficiency measured by how well they scale with input size
- Big O notation characterizes functions according to their growth rate by suppressing multiplicative constants and lower order terms (upper bound): e.g. $5n^3 + 3n$ is $O(n^3)$
- Constant time (or space) complexity expressed as $O(1)$
- $O(n)$ denotes a linear, $O(2^n)$ an exponential time algorithm
- Bubble sort is $O(n^2)$: n rounds of n comparisons and swaps
- Merge sort is $O(n \log n)$: log n rounds of linear merging

Complexity Theory

- Classifying problems according to their inherent difficulty; Determine the practical limits of what computers can do
- A complexity class is a set of problems of related difficulty:
  - P: Problems deterministically solvable in polynomial time
  - NP: Non-deterministically solvable in polynomial time, i.e. a solution to the problem can be verified in polynomial t.
- Clearly, P $\subseteq$ NP, NP containing many important problems; Hardest problems in NP are NP-complete (reduction…)
- Example for NP-complete: The subset sum problem ($\sum = 0$)
- P $\stackrel{?}{=}$ NP problem: Efficient check implies efficient solution?

Computability Theory

- Asks what kind of problems can be solved algorithmically
- A decision problem is a question with a yes-or-no answer
- A decision problem solvable by an algorithm is decidable
- Halting Problem: Given a description of a program and a finite input, decide whether the program finishes running
- An algorithm is required to terminate (i.e. in finite time)
- Alan Turing proved in 1936 that no general algorithm to solve the halting problem for all possible pairs can exist
- Proof by reduction: New algo. would solve undec. problem

Concepts Learned Today

- Abstraction
- Algorithm
- Bootstrapping
- Bottleneck
- Caching
- Complexity
- Pipelining
- Specification
- Transparency

Clip of Today

My favorite comment:
Then again, there's the Brute Force technique: Steal already sorted arrays from third world countries. O(1).

Barack Obama - Computer Science Question (1:25)

http://www.youtube.com/watch?v=k4RRi_ntQc8

Questions?