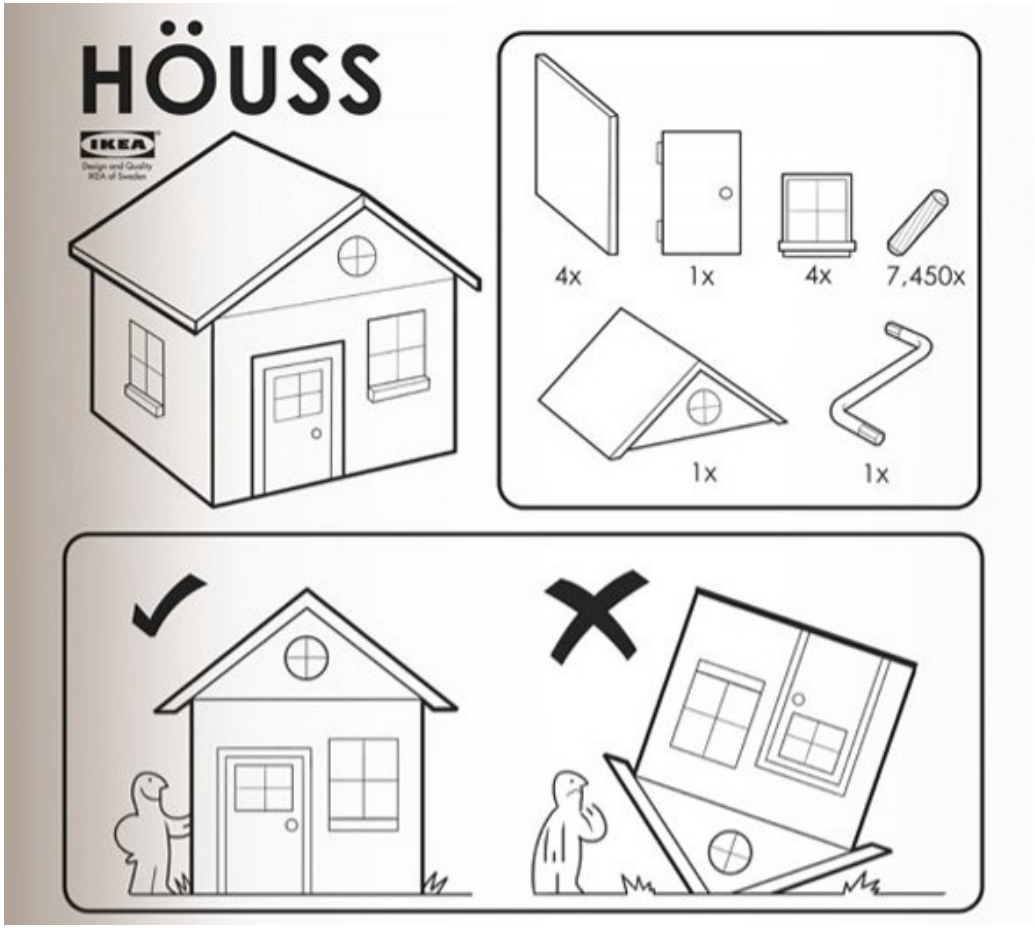
Instruction Set Architecture（ISA）

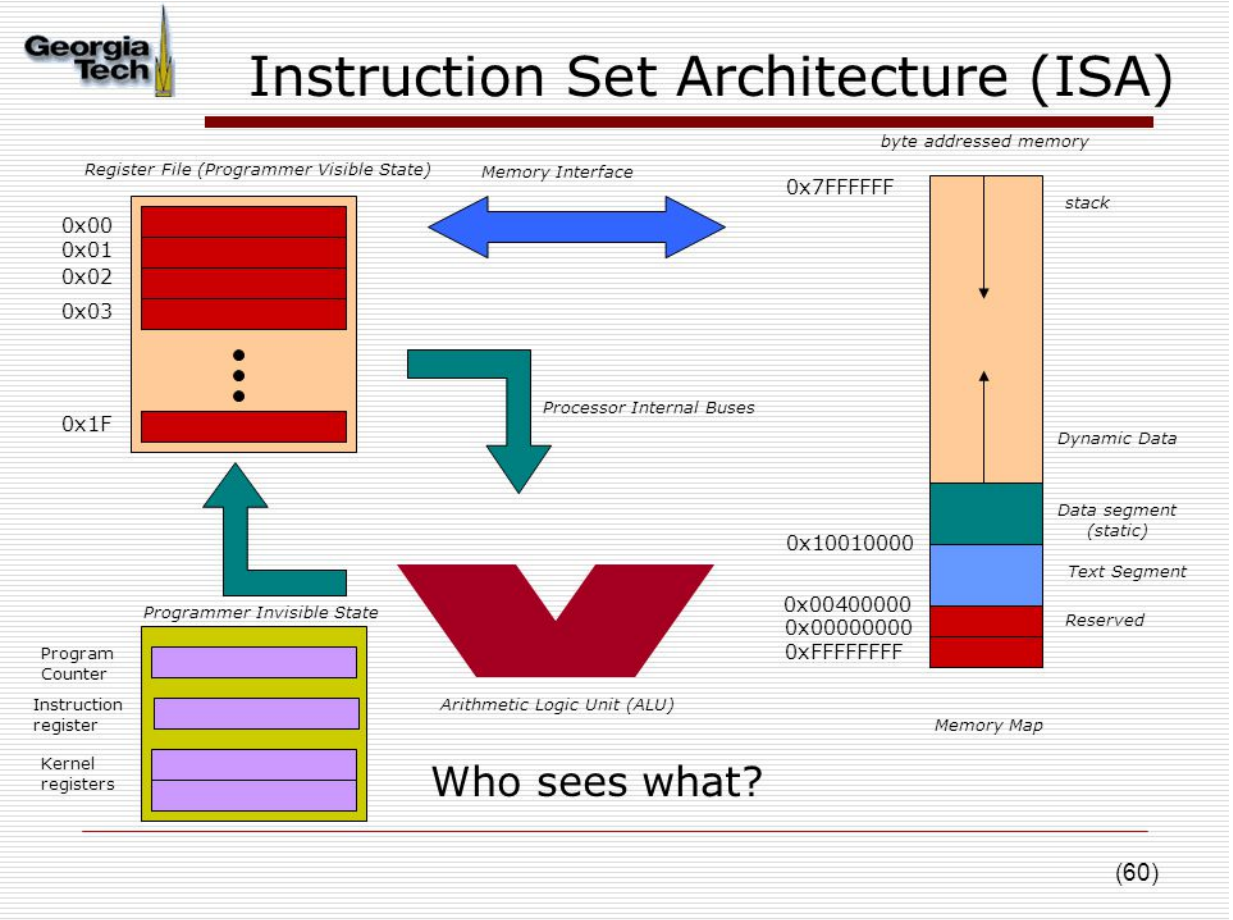
instructions



Instruction Set Architecture

- Also called (computer) architecture
- Implementation --> actual realisation of ISA
- ISA can have multiple implementations
- ISA allows software to direct hardware
- ISA defines machine language

ISA model


Georgia Tech
Instruction Set Architecture (ISA)
Register File (Programmer Visible State)
Memory Interface
byte addressed memory
0x00
0x01
0x02
0x03
0x1F
0x7FFFFFF
stack
Processor Internal Buses
Dynamic Data
Data segment (static)
0x10010000
Text Segment
0x00400000
0x00000000
0xFFFFFFFF
Reserved
Programmer Invisible State
Program Counter
Instruction register
Kernel registers
Arithmetic Logic Unit (ALU)
Memory Map
Who sees what?
(60)

ISA defines...

- Data types
- Memory (registers et. al.)
- Addressing modes
- Instruction set
- I/O

What is an ISA?

- x86 --> yes!
  - Intel implements x86
  - AMD implements x86
  - Yet, both have different designs!!!
- Therefore,
  processors with different designs
  (microarchitectures) can share the same ISA

Virtual Machines

- Java Bytecode, anyone???
  - Java programs are compiled to instruction set specific to the Java VM
  - Java VM translates bytecode to machine specific machine code
- Possible to implement one ISA one top of another one using such techniques

How to classify ISA?

- Based on *complexity*
  - Complex Instruction Set Computer (CISC)
  - Reduced Instruction Set Computer (RISC)
- Parallelism / Word size
  - VLIW (very long instruction word)
  - LIW (long instruction word)
  - EPIC (explicitly parallalel instruction computing)

CISC (complex)

- Single instruction can execute multiple operations (low level --> I/O, ALU, mem)
- CISC was defined *after* RISC was defined
  - But CISC came before RISC (*bizzaro!*)
  - Everything that is not RISC... is CISC

CISC (complex)

Designed to implement programming constructs such as:

- Proceduce calls
- Loops
- Array access
- Address lookups

... into a single instruction!!!

CISC (complex)

CISC is awesome.....

But...

There’s a catch...

Sometimes less complex instructions performed better...

Because programmers ‘overengineered’ (*typical!*)

CISC
and the return of the Superscalar

- Build superscalar implementations of CISC directly (native support)
- Advances in fast cache mediate frequent memory accesses
- Combine x86 instructions
- E.g. Pentium Pro and AMD K5
- Why didn’t it catch on???

  ... because everyone was using x86 RISC by then

Why did we take RISC?

- Reduced, as in, less complex than CISC
- Requires lesser cycles to execute
- Loose definition of instructions
  - simpler and smaller and general
- Has *more* (reduced set? no?) instructions

Stanford MIPS Experiment

- Combine RISC with VLSI (very large scale integration semiconductor technology)
- 32-bit everything
  - instruction, addressing (word-addressed)
- Load/Store
  - 32 general purpose registers
- Optimising compilers!!!

RISC

- Fixed-length instructions (mostly 32bit)
- Drawback? Code density.
- ARM, MIPS, RISC-V
  - short reduced instruction
  - instruction compression
- ARM… is the processor in your phones!

  You’re all carrying a RISC in your pockets!

What architecture is my desktop/laptop CPU?

RISC?

CISC?

What architecture is my desktop/laptop CPU?

It implements the best of both worlds!!!

Levels of parallelism

- Thread
- Task
- Data
- Memory
- Software

MPP
Massively Parallel Processing

- Large number of processors
- Simultaneously process
- Can be different computers
- e.g. Grid computing
- MPPAs – massively parallel processing arrays
- Used in supercomputers

  (this is not cluster computing)

Grid Computing

- Distributed system
- Non-interactive workloads
- Each node (can) perform a different task
- Nodes can be heterogenous
- Nodes are not physically coupled

GPU / GPGPU

- General Purpose computing on Graphical Processing Unit (GPGPU)
- Use GPU instead of CPU
- GPU is great at parallelisation
- Use multiple GPUs in a pipeline formation
- e.g. Nvidia CUDA
- e.g. Metal (Apple), Vulkan
- More later!!! (really later, in like, another lecture)

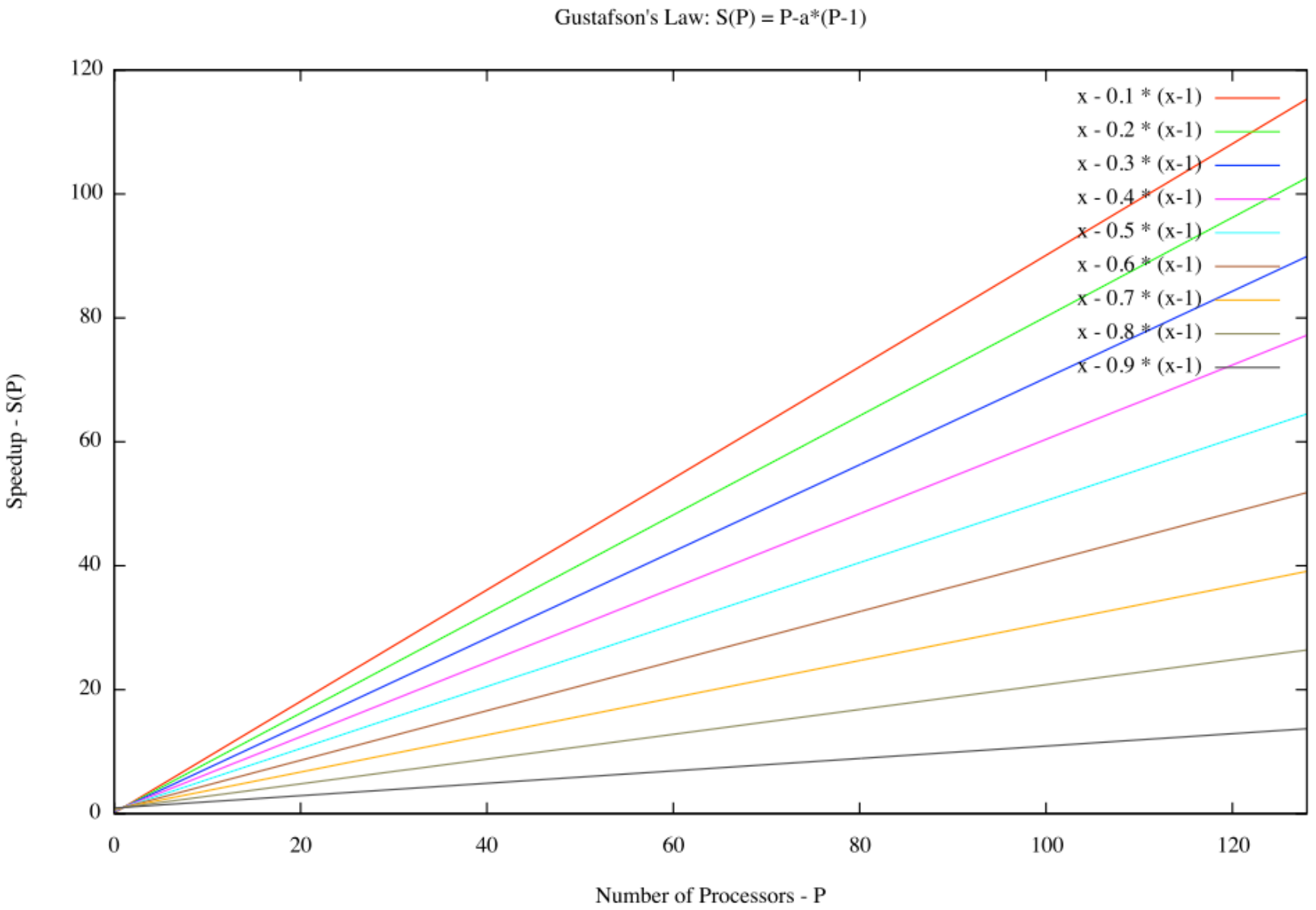
Vector Processor

- Instructions operate on vectors (1D) --> SIMD !!!
- Scalar processors operate on one item
- Awesome for numerical tasks
- GPUs are kind of like vector processors
- Intel x86
  - MMX, SSE, AVX
- We'll learn about SSE soon.

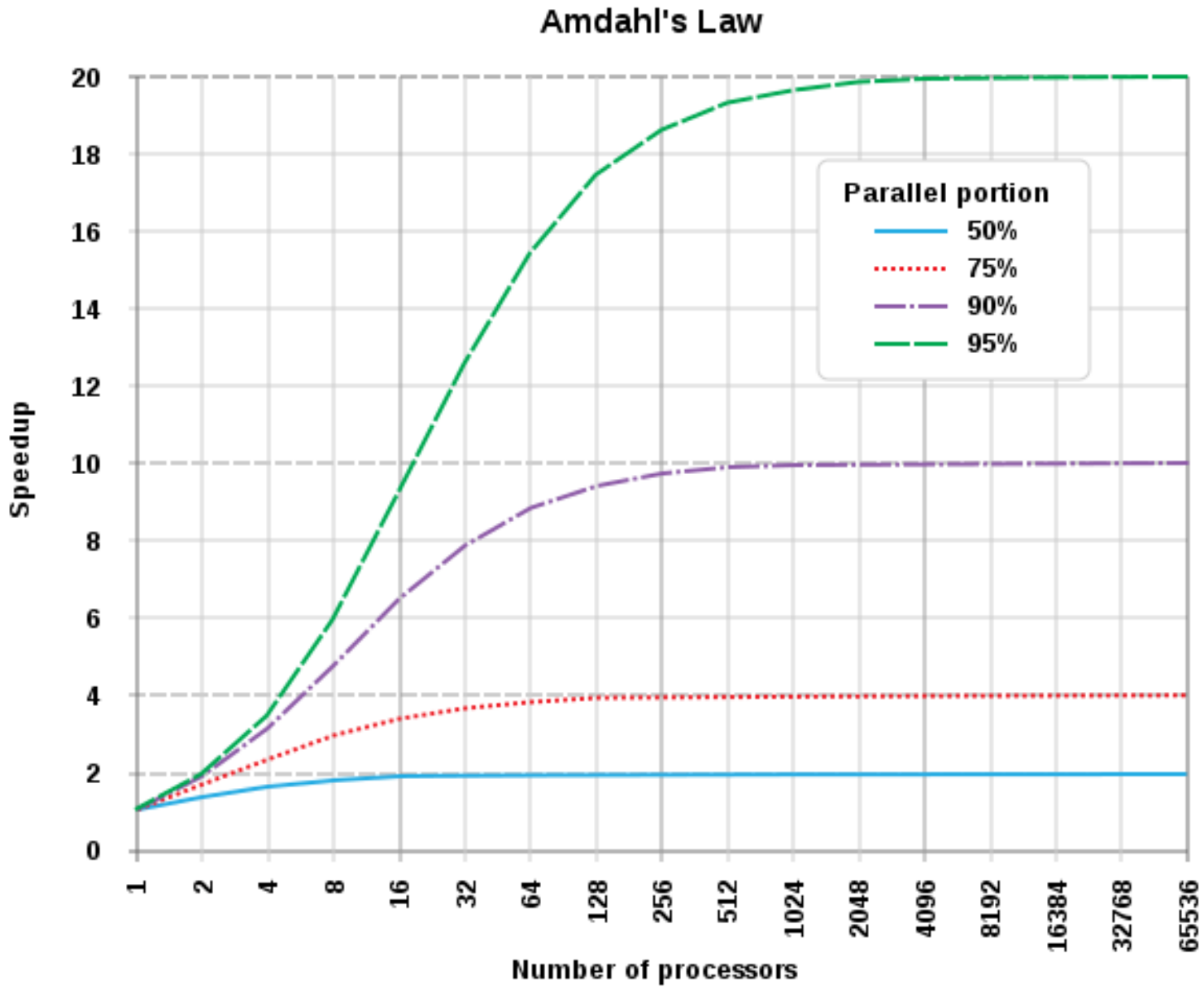
Gustafson’s & Amdahl’s Law

- A task executed by a system whose resources are improved compared to an initial similar system can be split into two parts:
  - a part that does not benefit from the improvement of the resources of the system;
  - a part that benefits from the improvement of the resources of the system.

Gustafson’s Law
fixed execution time



Amdahl’s Law
fixed workload



Parallel Programming Languages

- Actor model: Smalltalk, Erlang, Elixir
- Co-ordination: LINDA
- Dataflow: Joule
- Distributed: Julia
- Event-driven: Verilog, VHDL
- Functional: Haskell
- Logic: Prolog

Parallel Programming Languages

- Multi-threaded: C, C++, Java
- Object-oriented: C#, Java, Smalltalk
- Message passing: Rust

- Frameworks
  - Apache Hadoop, Apache Spark, CUDA,
  - OpenCL, OpenMP

Let’s discuss the assignment!!!

The assignment is *simple*.
Implement 1 **searching algorithm** and 1 **sorting algorithm** in non-parallel and parallel execution formats, and compare them.
Write a report on your findings.

Submission date: FRIDAY 23:59 16 March 2018 UTC.

How to submit: blackboard, upload as many times as you want