

# Programming Language Design

Dominic Orchard  
Computer Laboratory  
`dominic.orchard@cl.cam.ac.uk`

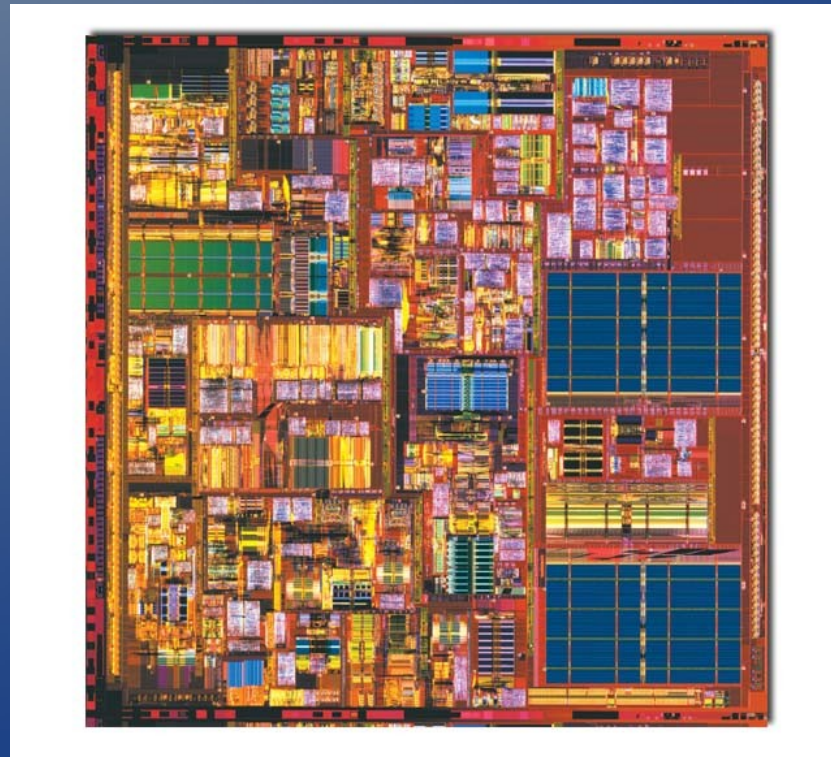
*1<sup>st</sup> of May 2009*

# What are programming languages?

- Computers: not autonomous
- We must instruct them
- Programming languages define a language of instructions for writing programs
- “Simpler” than natural language
  - More formally defined
  - Less ambiguity

# Processor

- Complex circuitry
- Controlled with a set of basic instructions



# Programming Language History

- 1940s - First electronic computers
- 1948 - Plankalkül
- 1957 – FORTRAN
- 1958 – LISP
- .....
- 1972 – C
- .....
- 1995 – Java

# Abstraction

- Programming languages abstract over processor instructions
  - Factor out detail
- Abstractions help us write bigger programs more quickly (with hopefully less mistakes!)
- Compiler
  - Translates programming language into processor instructions

# Syntax

- The symbols and structure of the language
- e.g. English
  - Latin alphabet + grammar marks
  - Word and sentence structure

# Semantics

- What does a program mean?
- Must be very precise
- A lot of deep mathematical work

# 4 Rs of Programming Language Design

- 'Riting
- Reading
- Reasoning
- Running

# 'Riting

- Programs must be “easy” to write
- Programmer understands what is happening
- Solve problems quicker
- Make less errors/mistakes
- Manage complexity

# Reading

- Easy to understand later
- Understandable by another programmer
- Manage complexity
  - Facebook ~400,000 lines of code (guess)
  - Windows XP ~ 40 million lines of code

# Reasoning

- Computer programs are everywhere
- Safety-critical applications
  - Trains, planes, pace-makers, surgical tools
- Reason about or prove the “correctness” of a program
- Compilers need to stop us doing stupid things

# Reasoning (Type Systems)

- $1 + 2$ 
  - $= 3$
  - “+” takes two numbers and gives a new number
- $1 + \text{cat}$ 
  - $= ???$
  - “cat” is of the wrong “type”
- Compiler should tell us: + should take two numbers, not a number and an animal

# Running

- Want programs to run fast – efficiently
- Language can affect what a compiler can do to make an efficient translation to processor instructions

# Summary

- Languages help us express programs to solve problems
- Much research in programming languages to ease:
  - Reading
  - Writing
  - Reasoning
  - Running
- Many languages suited for different tasks

# What I do...

- Develop new languages / extend current languages
- Ypnos: Language for scientific computing
- Help programmers write “parallel” programs for large simulations
- Language has special abstractions to help express these simulations
- Compiler able to split a simulation into parts to run in parallel

Thanks.