Compiler Design

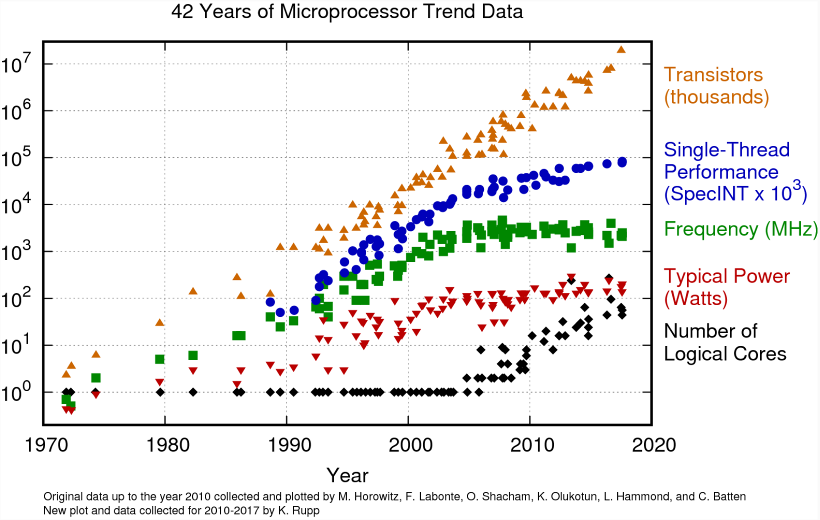
Lecture 22: Conclusions

Christophe Dubach

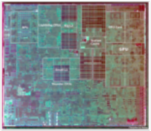
Winter 2022

Timestamp: 2022/04/07 20:08:00

Historical Data

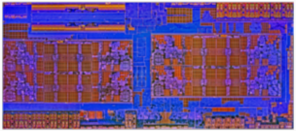


Era of Billion-Transistor Chips



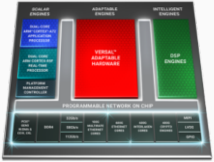
Apple A13
~8B transistors

Apple M1
~16B transistors



AMD EPYC Rome
~39B transistors

NVIDIA A100 Ampere
~54B transistors



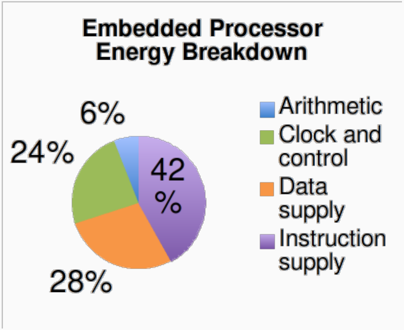
Xilinx Versal VP1802
~92B transistors

Inefficiency of General-Purpose Computing

Typical energy overhead for every 10pJ arithmetic operations:

- 70pJ on instruction supply
- 40pJ on data supply

Plus, only 59% of instructions are arithmetic!



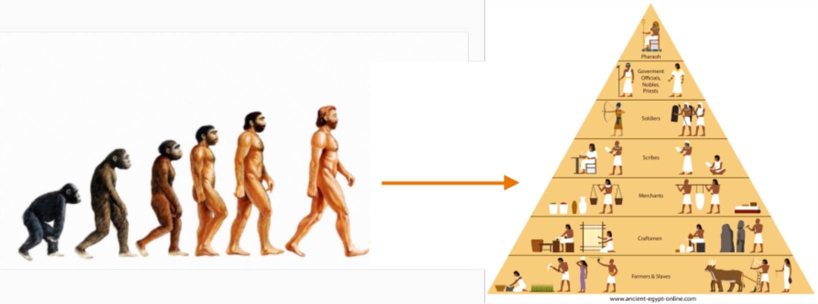
[source: Dally et al. Efficient Embedded Computing, IEEE'08]

Advance of Civilization

For humans, Moore's Law for scaling of the brain has ended a long time ago

- Number of neurons and their firing rate did not change significantly

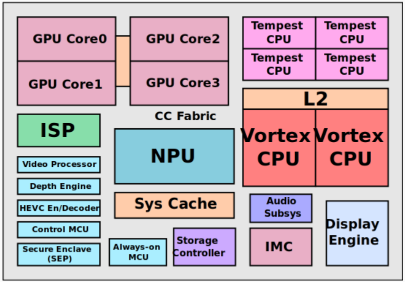
Remarkable advancement of civilization via specialization



source: https://en.wikichip.org/wiki/apple/ax/a12

Computers are Following the Same Path: Diverse Range of Integrated Functionalities

System on Chip



source: [https://en.wikichip.org/wiki/apple/ax/a12]

Modern SoCs integrate a rich set of special-purpose accelerators

- Speed up critical tasks
- Reduce power consumption and cost
- Increase energy efficiency

Specialization creates challenges for compilers!

Specialized architecture looks different from general pupose CPU

- coarse-grained specialized instructions: *e.g.* MxM
- memory hierarchy more complex to manage: local memories
- needs to detect pattern of code in the program: more complex form of instruction selections
- special optimizations might be needed, *e.g.* tiling of data to fit into small accelerator memory
- hardware might be highly parallel, *e.g.* GPUs with thousands of threads

Specialized hardware often require specialized languages:
Domain Specific Languages

- have you already used a DSL?
- plenty of others emerging, *e.g.* tensor algebra, neural networks, graph algorithms
- all these require compiler support

Big research question

Could we design one compiler to rule them all?

- What does the IR would look like?
- What about optimizations?
- General mechanism for finding pattern of code to accelerate?
- Can we deal with multiple front-ends?
- Can we automatically partition a program to run across different type of devices?
- How to detect and exploit parallelism?

What's next for you?

In this course, we have only scratched the surface of the world of compilers. Compilation is still a very active research field and there is plenty of development.

If you want to gain experience with industry compilers:

- For C like languages: LLVM
- For Java like languages: GraalVM / Truffle (from Oracle Labs)
- For JavaScript: V8

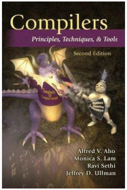
Hot compiler IRs:

- MLIR (related to LLVM)
- WebAssembly (virtual assembly for the web)

Courses you may also like:

- COMP 764 : "High-Level Hardware Synthesis" (TBD, Winter 2023)
- ECSE 427 / COMP 310 : Operating Systems
- COMP 409 : Concurrent Programming

What to read next:



The "Dragon book":
Compilers: Principles, Techniques, and Tools
Alfred Aho*, Monica Lam, Ravi Sethi, Jeffrey Ullman*

*ACM Turing Award Winners, 2020

“Compiler” Conferences

- ACM/IEEE International Symposium on Code Generation and Optimization (CGO)
- ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)
- ACM SIGPLAN International Conference on Compiler Construction (CC)
- International Conference on Parallel Architectures and Compilation Techniques (PACT)
- International Conference on Compilers, Architectures, and Synthesis for Embedded Systems (CASES)
- ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)
- International Conference on High Performance and Embedded Architectures and Compilers (HiPEAC)

Research in my group (COMP 400, ECSE 498, SURE/SURA)

- Parallel programming abstractions
- Rewrite-based optimizations
- High performance code generation
- High-level hardware synthesis

Looking for a job related to compilation?

- `https://github.com/mgaudet/CompilerJobs`

The end