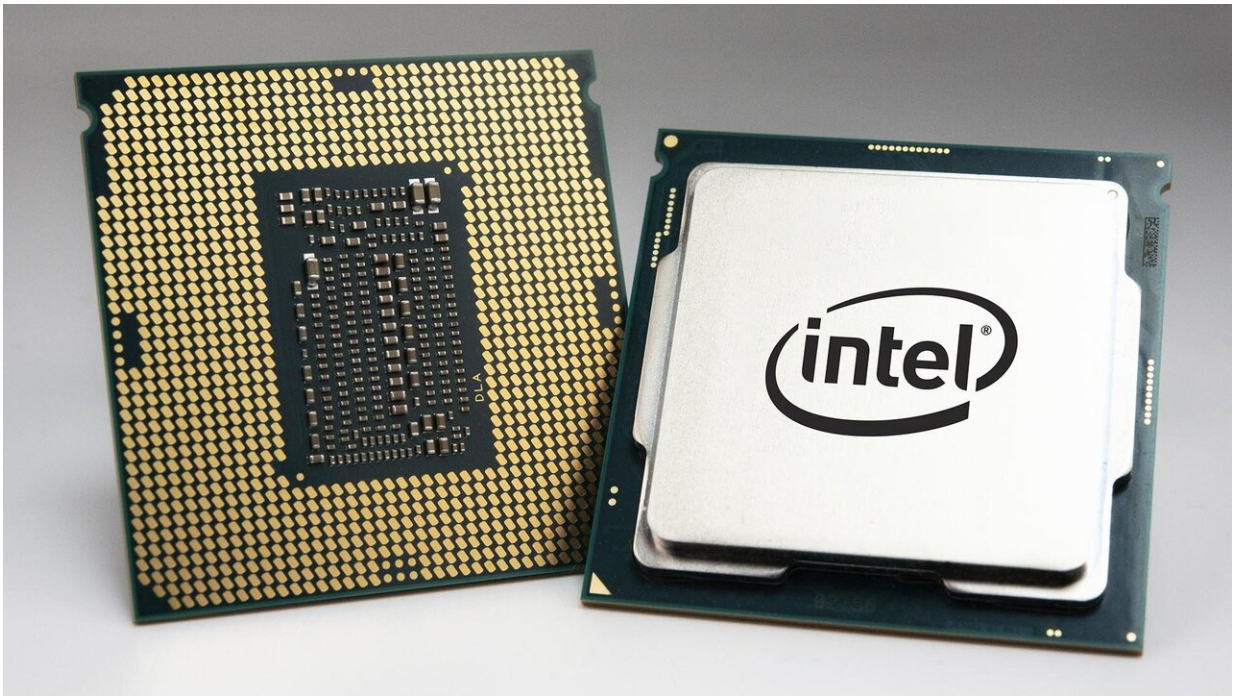
A tool for navigating complex computer instructions

April 19 2021, by Rachel Gordon



Credit: Intel

We've come a long way since Intel introduced the first microprocessor in 1971. Their 4004 held 2,300 transistors, with today's best chips exceeding billions, harnessing more and more power since their birth.

But every time Intel releases a new computer chip, it's a costly investment, as they need to add new instructions of computer programs

that tell it which data to process and how to process it. These are things a user doesn't see, but that power tasks like image processing, machine learning, and video coding.

However, the programs that process this new information, called compilers, can't always use these more complex instructions. The burden then often falls on expert developers to do more of the work by hand, and to perform error-prone and cumbersome tasks like writing assembly code.

Scientists from MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) came up with a way to better navigate the complexity of supporting these instructions. Their tool "VeGen" (pronounced "vegan") automatically generates compiler plugins to effectively use more complicated instructions.

CSAIL Ph.D. student Yishen Chen says that VeGen takes in the same documentation that Intel gives to software developers, and automatically generates a compiler plugin that lets the compiler exploit something called non-Single Instruction Multiple Data (SIMD), which are instructions that are more complicated to accelerate a given user-supplied program.

"Without VeGen, compiler engineers have to read the documentation and manually modify the compiler to use these instructions," says Chen, an author on a new paper about VeGen. "The problems here are that this is still manual, and current compiler algorithms are not effective at using these instructions."

Instruction methods

Most processors use math-based instructions that allow you to do something like "A= B+C."

Processors also support something called vector instructions, which are instructions that do multiple but identical operations at once, such as "A1=B1+C1 and A2=B2+C2." These are both considered more traditional "SIMD" instructions.

"Non-SIMD" instructions are more complicated, but even more powerful and efficient, such as instructions that perform both additions and subtractions simultaneously. Chen says that VeGen is mostly motivated by instructions that don't fit the SIMD model, in one way or another.

Think of the whole process like a restaurant:

- The programmer is the celebrity chef who thinks of a program (a dish)
- The compiler is the sous chef who takes the program (the dish) and creates instructions to be executed (recipe) knowing the resources available in the kitchen
- The processor is the kitchen
- The instructions (the recipe) are executed (prepared) by the line cooks taking advantage of different equipment in the kitchen
- A new processor design that adds new capabilities to the processor is the remodeling of the kitchen, or the addition of new kitchen equipment.

If the sous chef and his team don't know how to use the new equipment, the restaurant owners who spend all the money remodeling the kitchen will not be happy.

"With the advent of complex instructions, it's become hard for compiler developers to keep code generation strategies up-to-date in order to harness the full potential supported by the underlying hardware," says Charith Mendis, professor at the University of Illinois at Urbana-

Champaign, an author on a paper about the tool. "VeGen's approach to building code generator generators alleviates this burden by automatically generating parts of the compiler responsible for identifying code sequences that can exploit new hardware features. We hope that VeGen's approach to building compiler components will lead to more sustainable and maintainable compiler infrastructures in the future."

Initial results showed that, for example, on select video coding kernels, VeGen could automatically use non-SIMD vector instructions and get speedup from 27 percent to 300 percent.

"Putting all the Intel instruction manuals together is more than one foot wide, going into thousands of pages," says MIT professor Saman Amarasinghe, an author on the paper about VeGen. "Normally, the compiler writer has to pour over the fine details of instruction changes, spread over hundreds of pages, but VeGen totally bypasses the tedious work."

"As hardware becomes more complicated to accelerate compute-intensive domains, we believe VenGen is a valuable contribution," says Chen. "The long-term goal is that, whenever you add new features on your hardware, we can automatically figure out a way —without having to rewrite your code—to use those hardware accelerators."

Chen wrote the paper alongside Mendis, and MIT professors Michael Carbin and Saman Amarasinghe. They will present the paper virtually at the Architectural Support for Programming Languages and Operating Systems (ASPLOS) conference in April.

Provided by Massachusetts Institute of Technology

Citation: A tool for navigating complex computer instructions (2021, April 19) retrieved 5 February 2023 from https://techxplore.com/news/2021-04-tool-complex.html

This document is subject to copyright. Apart from any fair dealing for the purpose of private study or research, no part may be reproduced without the written permission. The content is provided for information purposes only.