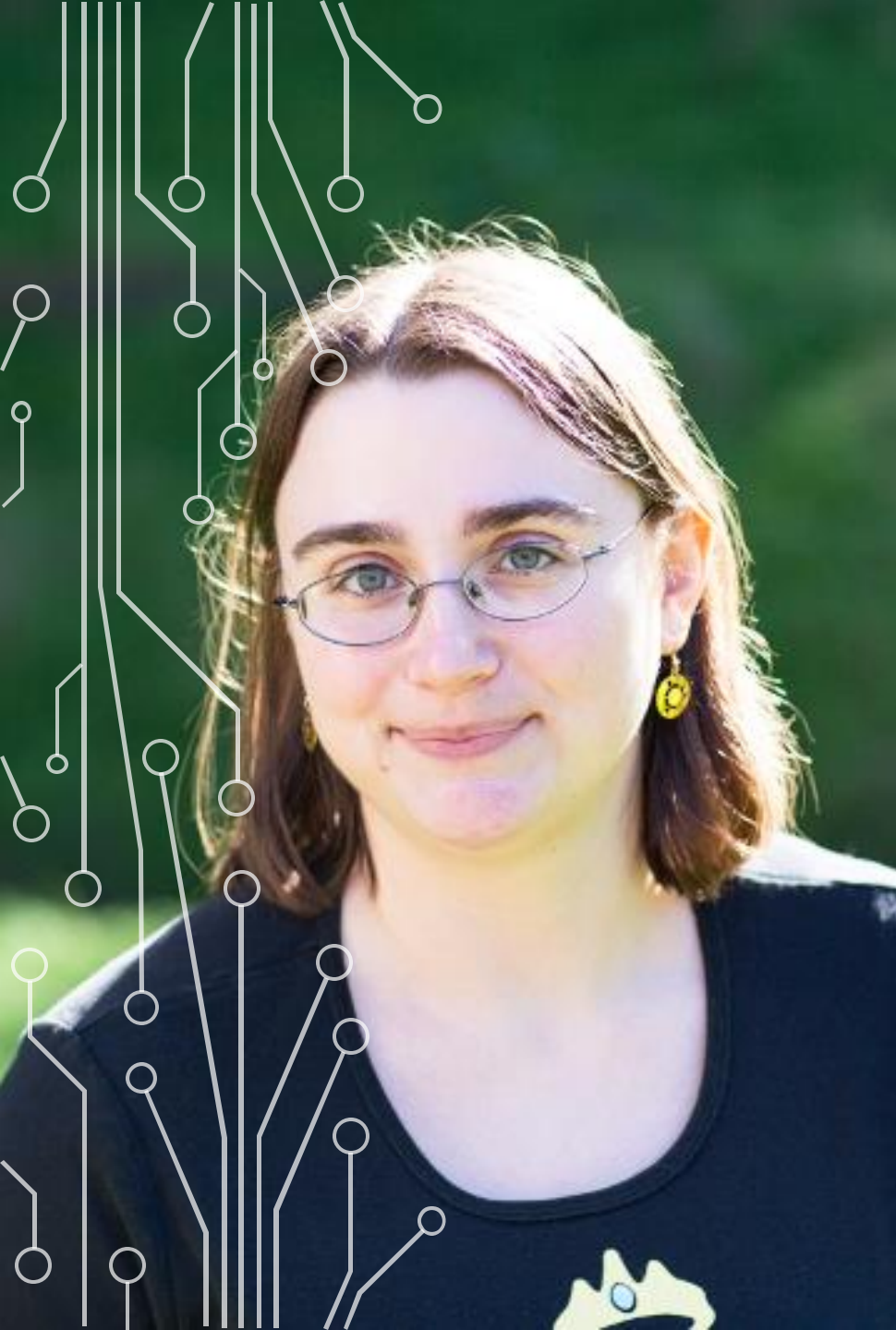




DEVELOPING OPEN
SOURCE
SOFTWARE FOR
VARIOUS
HARDWARE
ARCHITECTURES

ELIZABETH K.
JOSEPH, IBM
@PLEIA2

HASHNODE OPEN
SOURCE
SYMPOSIUM,
OCTOBER 2021



ELIZABETH K. JOSEPH

- Linux Systems Administrator turned Developer Advocate for IBM Z
- Author of books on Ubuntu Linux and OpenStack
- Contributor to open source communities for 20 years
- @pleia2 on Twitter, Instagram

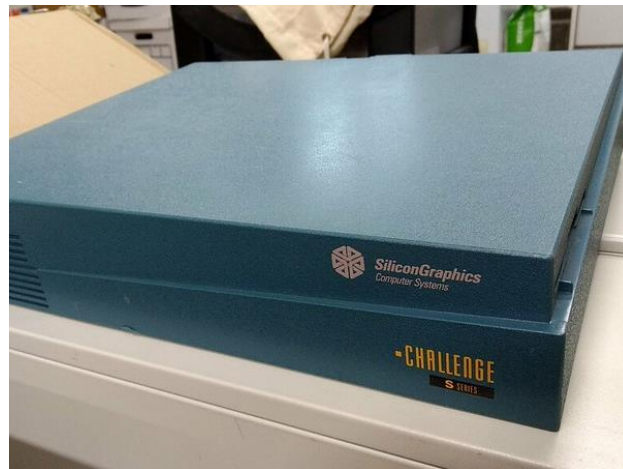


AND I WAS VERY
FOCUSED ON
CLOUDS



LET'S GO BACK IN TIME

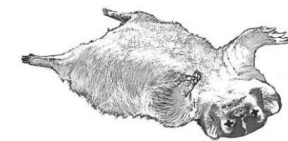
ONCE UPON A TIME, THERE WERE MANY ARCHITECTURES



Techie Zombie Humor Inside ~ As Seen On Slashdot!

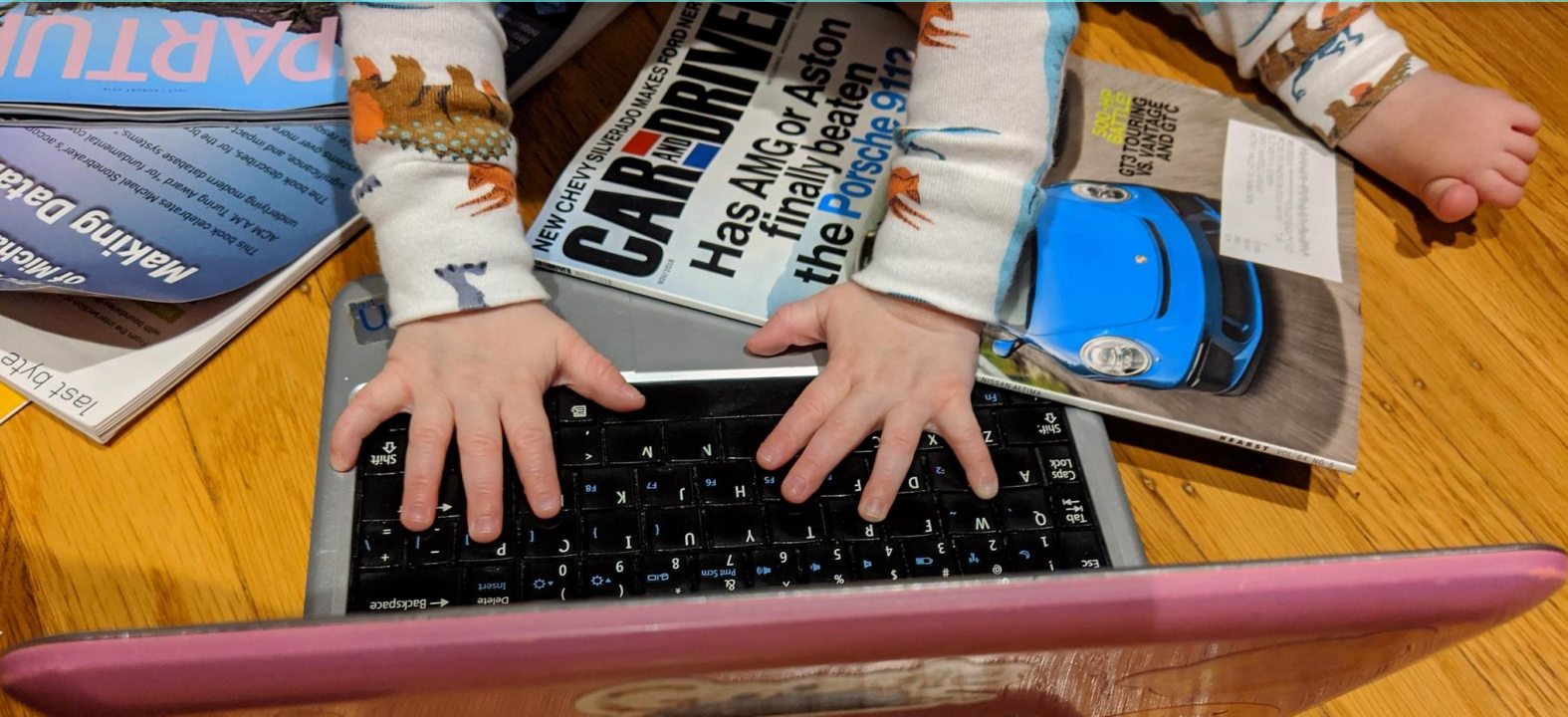
INSTALLING LINUX

On A Dead Badger



And Other Oddities

Written by Lucy A. Snyder
Illustrated by DE Christman and Malcolm McClinton



THEN WE KINDA
CONSOLIDATED
ON X86



**NOW THERE ARE
MANY AGAIN!**

KIND OF, THEY NEVER REALLY WENT AWAY ;)



WHY AM I HERE TALKING ABOUT ARCHITECTURES?

- I [pre-COVID] worked out of the IBM Silicon Valley Lab
- It's in the cows and jack rabbits part of San Jose
- And it has a big, underground, datacenter





IBM Z – S390X / ZARCHITECTURE

A close-up photograph of a microchip with a gold-colored surface and a grid of small square components. The text "IBM POWER10" is overlaid in white, bold, sans-serif font.

IBM POWER10



IBM POWER /
OPENPOWER

AND SO MUCH ARM

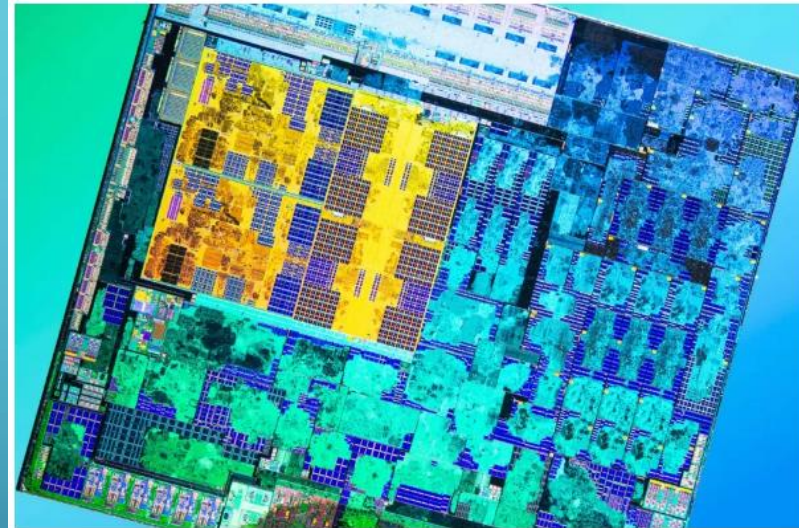


AMD: We Stand Ready to Make Arm Chips

By Paul Alcorn 27 days ago

The ARMy grows

[f](#) [t](#) [i](#) [p](#) [r](#) [e](#) [m](#) Comments (28)



(Image credit: Fritzens Fritz)

AMD's CFO Devinder Kumar recently commented that AMD stands ready to manufacture Arm chips if needed, noting that the company's customers want to work with AMD on Arm-based solutions. Kumar's remarks came during last week's Deutsche Bank Technology Conference, building on comments from AMD

iPhone



YOUR PHONE - ARM



RASPBERRY PI - ARM

COMPILING SOFTWARE 101 – ARCHITECTURE

Today the commodity architecture is 64-bit x86, based on instructions developed in the early 1980s.

As a result, most developers don't pay much attention to architecture! But we are seeing an increasing need to do so as non-x86 architectures become more common.


COMPILING SOFTWARE 101 – CODE

- At the lowest levels, classic* computing still only understands 0 and 1. That's what all those billions of tiny transistors are doing.
- Compilers and interpreters take human-readable code that you write and convert it to something the computer can understand, ultimately a series of 0s and 1s.
- The code you see is just the first step in the process as far as the computer is concerned.

* What is beyond Classic Computing? Quantum!

COMPILING SOFTWARE 101 – OPEN SOURCE

When something is "open source" you have access to the human-readable code, it's available in the open.



You then compile that code to create a binary. This code must be compiled for the respective architecture you're targeting since it needs to be built for that CPU hardware (x86, s390x, ARM, Power, etc).

HIGHER VERSUS LOWER-LEVEL LANGUAGES

It has very little to do with how "hard" the language is, and more to do with how much abstraction is between your code and the hardware.

Lower-level is closer to the hardware, and may have optimizations: Assembler, C, C++

Higher-level is further from the hardware, and often doesn't care where it's run: Python, Node.js

WHAT'S A DEVELOPER TO DO?

Well, you could do nothing, carry on!

Especially if you're working with higher-level languages or SDKs for your platform, you may not run into issues (this is often the case with mobile app development).

WHAT'S A DEVELOPER TO DO?

- Learn more about architecture-specific components of your language
 - Be mindful about your usage and don't use them unless you have a specific reason to do so
 - Avoid making assumptions about hardware-specific things like pointer sizes or byte ordering
 - Document the usage, so it's easier for anyone who may wish to port your code in the future

WHAT'S A DEVELOPER TO DO?

- Avoid "tricks" with CPU-specific instructions and caching
 - Some developers over-optimize their code and drop to Assembler
 - Modern compilers are already pretty smart!
 - Today's tricks may not even work on tomorrow's compiler, or x86 system
- Don't make assumptions about hardware enumeration or memory regions

WHAT'S A DEVELOPER TO DO?

- Try running your code on another architecture!
 - A Raspberry Pi 4 (ARM) kit with 4G of RAM will run you about \$100, and several major Linux distributions will run on it
 - You can sign up for an s390x Linux virtual machine for free for 120 days with in the IBM LinuxONE Community Cloud: <https://linuxone.cloud.marist.edu/>

RESOURCES

IBM Z

Developer resources for building your open source app for Linux on IBM Z

<https://developer.ibm.com/blogs/developer-resources-for-building-your-open-source-app-for-linux-on-ibm-z-and-linuxone/>

RESOURCES

POWER

Learning path: Port your open source applications to Linux on Power

<https://developer.ibm.com/series/learning-path-port-your-app-to-lop/>

RESOURCES

Raspberry Pi

Documentation (including Technical Information about the CPUs)

<https://www.raspberrypi.com/documentation/>

RESOURCES

Apple M1

Developer Transition Kit (DTK)

<https://developer.apple.com/programs/universal/>

GOING DOWN THE RABBIT HOLE

- Learn about the following key terms
 - RISC verses CISC
 - ISA (Instruction Set Architecture)
 - Conditional execution
 - Hardware registers (interface between hardware and software)
 - Hardware threads and hyperthreading
 - CPU cache and the Translation Lookaside Buffer (TLB)
 - Endianness (memory ordering, big- verses little-endian)
 - ...probably a lot more, but this is a good start!

CONTACT

Elizabeth K. Joseph

@pleia2 on Twitter and Instagram

lyz@princessleia.com | lyz@ibm.com