



A Arquitetura RISC-V, suas Implementações e o Software Livre

Carlos Eduardo - Embaixador RISC-V / Arquiteto de Cloud, Red Hat

Quem sou eu?



Carlos Eduardo

carlosedp

Check my articles on
<https://medium.com/@carlosedp> and
support me on
<https://www.patreon.com/carlosedp>

[Edit profile](#)

182 followers · 49 following · 153

RISC-V Ambassador / Red Hat

Sao Paulo/Brazil

me@carlosedp.com

twitter.com/carlosedp

→ Embaixador da RISC-V International

→ Arquiteto de Cloud na Red Hat

→ Projetos pessoais em multi-arquiteturas

- Artigos sobre Kubernetes em ARM
- Iniciei o suporte a tecnologias de containers como Docker para RISC-V
- Criador do projeto *riscv-bringup* para suportar aplicações cloud em RISC-V
- Projetos similar para arquitetura OpenPOWER
- Venho usando Linux em diversas plataformas, desde o RISC-V Kendryte K210 até o Sony PS3
- Contribuições a projetos open source para FPGAs
- Palestras na Kubecon, RISC-V Summit, Qcon.

<https://github.com/carlosedp>

<https://carlosedp.medium.com>

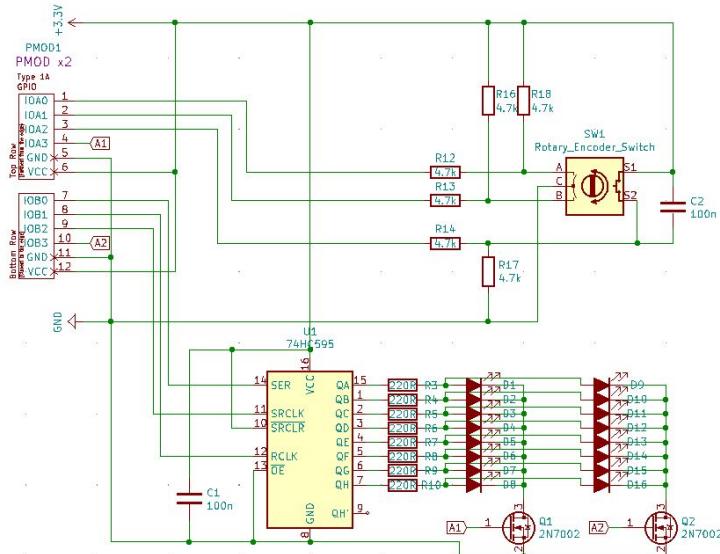
Hardware Open Source

É todo Hardware na qual o **design** é **público** e **disponível** para que qualquer pessoa possa **estudar**, **modificar**, **distribuir** e **vender** este design ou o hardware baseado nele.

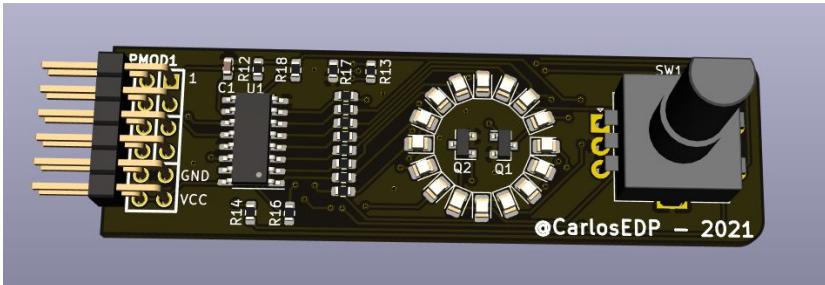
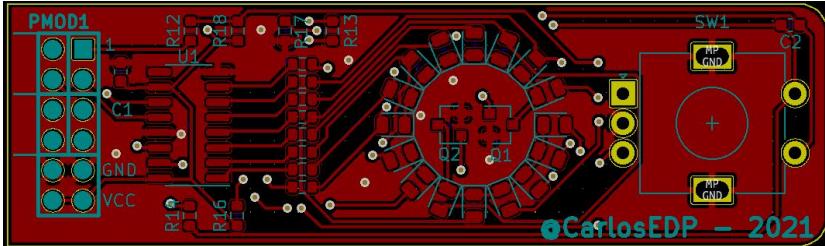
Fonte: Open Source Hardware (OSHW) Statements of Principles 1.0
<https://www.oshwa.org/definition/>

Exemplo de design Open Source

✓ Esquemático



✓ Placa de Circuito Impresso



Todos disponíveis em um repositório público

ISA - Instruction Set Architecture

ISA é o Design de Instruções para determinado processador, ele define:

- Interface entre o hardware e o software
 - Como um programa em uma linguagem (C por exemplo) envia instruções para o processador executar
- Definição de instruções para o compilador
 - O compilador gera instruções para o ISA alvo
- ISA é um padrão
 - É um conjunto de regras que define como o processador executa instruções
 - Pode ser aberto, com definições públicas (como RISC-V, Power ISA, SPARC) ou proprietárias e fechadas (como ARM, x86).

RISC-V - Um ISA aberto

Iniciado em 2010 pela Universidade da Califórnia Berkeley

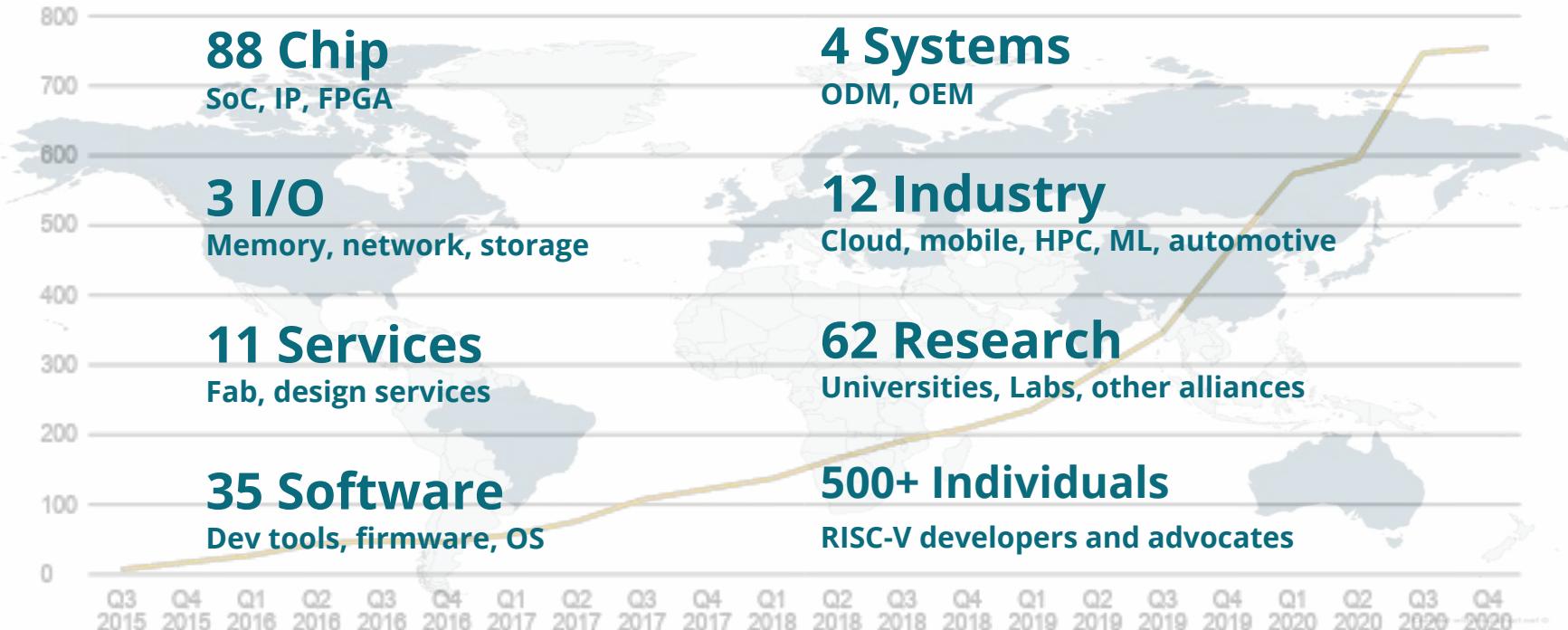
- Por que RISC?
 - RISC significa "Reduced Instruction Set Computer", um conjunto reduzido de instruções
- Por que V?
 - Esta é a quinta ISA da Universidade de Berkeley
- Por que aberto?
 - Pois suas especificações são abertas e públicas podendo ser utilizadas por quaisquer entidades sem custo ou licenciamento.

Comunidade



Mais de 1,000 RISC-V Membros

em mais de 50 países



Quais as diferenças do RISC-V?

- É um design novo
 - Utiliza um conjunto menor de instruções em comparação a outras arquiteturas
 - Possui uma separação clara entre instruções privilegiadas e não-privilegiadas
 - Construída com base em experiências anteriores com outras arquiteturas
- Arquitetura modular construída para ser extendida
 - Possui um conjunto base de instruções
 - É estendida com instruções padrão para funções mais comuns (multiplicação, manipulação binária, etc)
 - Desenhada para todos os tamanhos, desde microcontroladores até supercomputadores.
- Estável
 - O conjunto de instruções base é congelado e garante compatibilidade futura
 - Licenciado sob a licença *Creative Commons Attribution 4.0*
(<https://github.com/riscv/riscv-v-spec/blob/master/LICENSE>)

O que significa um ISA "Open"?

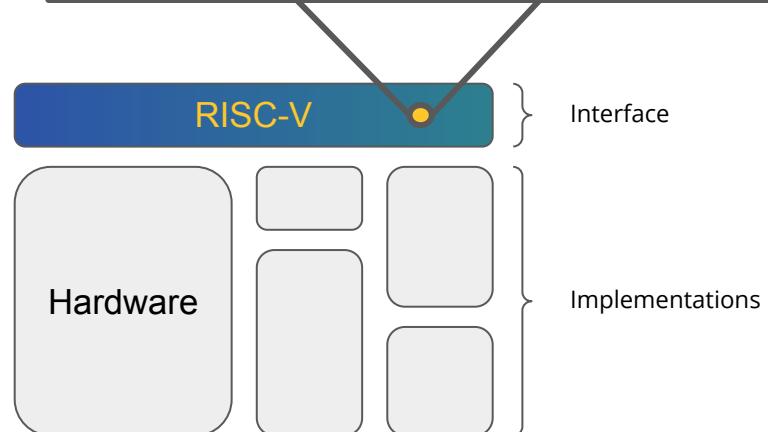
Pode ser utilizado para criação de processadores livremente sem necessidade de licenciamento por uma empresa ou entidade detentora da licença.

- Permite
 - Uso Comercial
 - Modificação
 - Distribuição
 - Uso Privado
- Não Permite
 - Responsabilidade
 - Propriedade das marcas registradas
 - Uso para patentes
 - Garantia
- Seu processador pode ser livre e aberto ou proprietário e fechado

O que é o RISC-V

- É um conjunto de instruções (ISA)
- Modular
- RV32I: Instruções 32bit
 - Possui menos de 50 instruções
- RV64I: Instruções 64bit
 - Pode ser estendida com instruções comprimidas, manipulações de vetor, manipulação binária, multiplicação, instruções atômicas e mais.

Name	Description	Version	Status
RV32I	Base Integer Instructions, 32 bit	2.0	Final
RV32E	Base Integer Instructions, 32 bit, embedded	1.9	Open
RV64I	Base Integer Instructions, 64 bit	2.0	Final
RV128I	Base Integer Instructions, 128 bit	1.7	Open
Q	Standard Extension Quad-precision Floating Point	2.0	Final
L	Standard Extension Decimal Floating Point	0.0	Open
C	Standard Extension Compressed Instructions	2.0	Final
B	Standard Extension Bit Manipulation	0.36	Open
M	Standard Extension Integer Multiply and Divide	2.0	Final
A	Standard Extension Atomic Instructions	2.0	Final
F	Standard Extension Single-precision Floating Point	2.0	Final
D	Standard Extension Double-precision Floating Point	2.0	Final
J	Standard Extension Dynamically Translated Languages	0.0	Open
T	Standard Extension Transactional Memory	0.0	Open
P	Standard Extension Packed SIMD Operations	0.1	Open
V	Standard Extension Vector Operations	0.2	Open
N	Standard Extension User Level Interrupts	1.1	Open



RISC-V Base e Extensões Padrão

- Extensões padrão
 - **M**: Multiplicação/Divisão de inteiros
 - **A**: Instruções de memória atômicas
 - **F, D, Q**: Instruções de ponto-flutuante de precisão simples, dupla e quadrupla
 - **G**: Conjunto das instruções **IMAFD**
 - **C**: Compressão de instruções para menor uso de memória em sistemas embarcados
- O Alvo das maiores distribuições Linux é o **RV64G**
- Grande parte das distribuições Linux já suporta a arquitetura riscv64.
- Suporte ainda se limita aos pacotes chamados "unstable" ou "experimental"

Hardware RISC-V

SiFive Freedom HiFive1

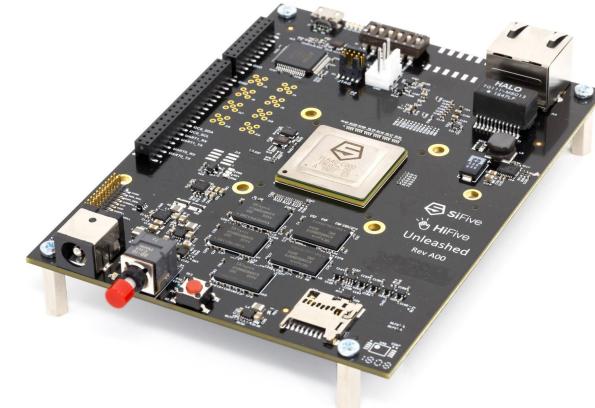
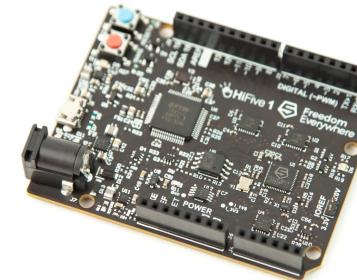
Implementa a ISA RV32i com foco em microcontroladores embarcados

- Utiliza o core FE310 de 320Mhz, 16MB ROM e 16Kb RAM;
- Suporte nos sistemas operacionais RealTime Zephyr, FreeRTOS;

SiFive Freedom Unleashed

Implementa o ISA RV64GC para Linux

- Utiliza o Freedom FU540 SoC, um processador Quad-Core e 8GB RAM
- Possui suporte 100% upstream no Kernel Linux

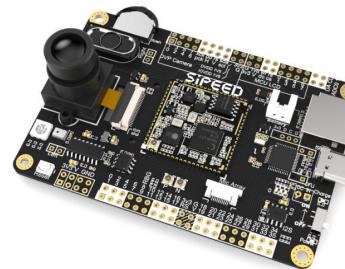


Hardware RISC-V (2)

Kendryte K210

Implementa a ISA RV64GC porém com MMU em versão draft.

- Utiliza o dual-core K210 com de 400Mhz, 8MB RAM
- Suporte nos sistemas operacionais RealTime Zephyr e FreeRTOS, Linux (sem MMU), Arduino;



Microchip Icicle (Polarfire SoC)

Implementa o ISA RV64GC para Linux

- Utiliza o quad-core FU740 a 667Mhz e 2GB RAM e possui um FPGA de 250k LE.
- Em processo de upstream no Kernel Linux



Hardware RISC-V (3)

SiFive Unmatched

Implementa a ISA RV64GC, usa padrão mini-ITX e é considerado o primeiro PC RISC-V

- Utiliza o quad-core FU740 de 1.2Ghz, 16GB RAM
- Em processo de upstream no Kernel Linux



BeagleBoard BeagleV

Implementa o ISA RV64GC para Linux

- Utiliza o StarFive 7100 quad-core com 4GB RAM
- Em processo de upstream no Kernel Linux



Ambas as placas possuem foco de desenvolvedores Linux

Cores RISC-V Abertos

Podem ser testados/modificados e carregados em FPGAs

VexRiscv - Core 32 bits de alta performance. Pode ser integrado com diversos dispositivos (rede, memória, UART, SDCard). Escrito em SpinalHDL e integrado em LiteX.

Rocket - Core 32/64 bits da SiFive escrito em Chisel. Base para as placas SiFive HiFive1 e Unleashed.

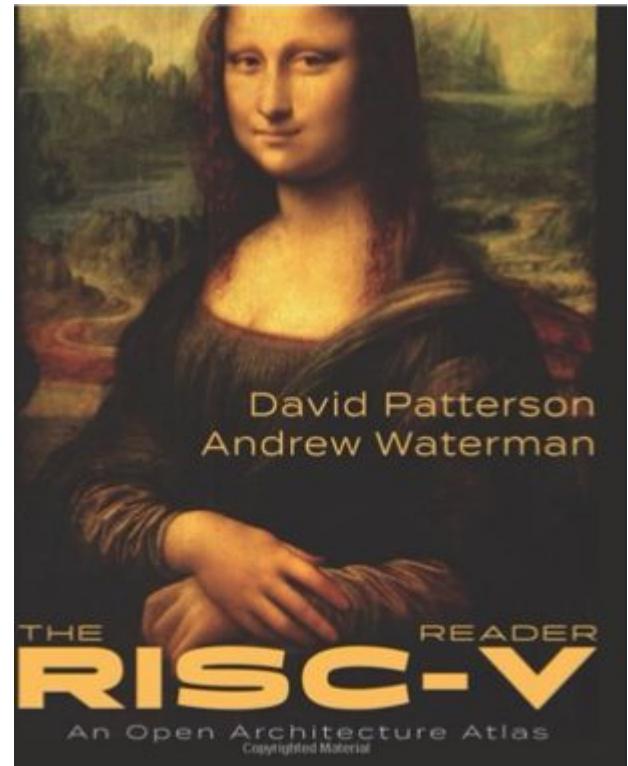
DarkRISCV - Core 32 bits escrito em Verilog pelo brasileiro Marcelo Samsoniuk

Mais em: <https://riscv.org/exchange/cores-socs/>

RISC-V - Livro Gratuito

Guia Prático RISC-V:
Atlas de Uma
Arquitetura Aberta

<http://riscvbook.com/portuguese/>



RISC-V e o Linux

- Suporte inicial no Kernel 4.15
- Suportado pelas maiores distribuições:
 - Fedora
 - Debian
 - Ubuntu
 - OpenSUSE
 - Gentoo
 - OpenEmbedded / Yocto
 - BuildRoot
- 100% funcional no emulador QEmu
- Suportado em diversas placas com processadores RISC-V

RISC-V e o Linux

```
.-/+00SSSS00+/-.  
`:+SSSSSSSSSSSSSSSSSSS+:`  
-+SSSSSSSSSSSSSSSSySSSS+-  
.0SSSSSSSSSSSSSSSSdMMMNySSSSO.  
/SSSSSSSSSShdmNNmyNNMMhSSSSSS/  
+SSSSSSSSSShydMMMMMdddySSSSSSS+  
/SSSSSSSShNMMyhyyyyhNMNhSSSSSSS/  
.SSSSSSSSdMMNhSSSSSSSSSShNMMdSSSSSS.  
+SSShhyNMNySSSSSSSSSSyNMMMySSSSSS+  
0SSyNMMNyMMhSSSSSSSSSSSShmmhSSSSSSO  
0SSyNMMNyMMhSSSSSSSSSSSShmmhSSSSSSO  
+SSShhyNMNySSSSSSSSSSyNMMMySSSSSS+  
.SSSSSSSSdMMNhSSSSSSSShNMMdSSSSSS.  
/SSSSSSSShNMMyhyyyyhdNMNMhSSSSSS/  
+SSSSSSSSdmydMMMMMdddySSSSSSS+  
/SSSSSSSSSShdmNNNmyNNMMhSSSSSS/  
.0SSSSSSSSSSSSSSSSdMMNySSSSO.  
-+SSSSSSSSSSSSSSSSySSSS+-  
`:+SSSSSSSSSSSSSSSSS+:`  
.-/+00SSSS00+/-.  
  
root@Ubuntu-riscv64:~# lscpu  
Architecture: riscv64  
Byte Order: Little Endian  
CPU(s): 4  
On-line CPU(s) list: 0-3  
Thread(s) per core: 1  
Core(s) per socket: 4  
Socket(s): 1  
root@Ubuntu-riscv64:~#
```



<http://bit.ly/riscvtracker>

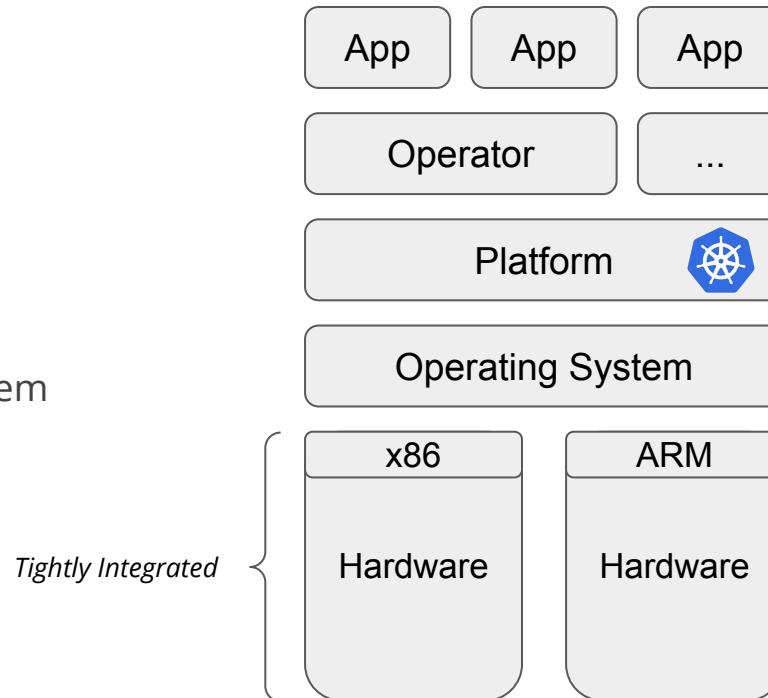
```
[ 0.405622] Run /init as init process  
Mounting filesystems  
none on / type rootfs (rw)  
devtmpfs on /dev type devtmpfs (rw,relatime)  
proc on /proc type proc (rw,relatime)  
sysfs on /sys type sysfs (rw,relatime)  
Saving random seed: [ 0.524121] random: uninitialized urandom read (512 bytes read)  
OK
```



```
BusyBox v1.31.1 (2020-02-26 14:47:57 EST) hush - the humble shell  
Enter 'help' for a list of built-in commands.
```

O que vem em seguida?

- O stack de software open source vem crescendo no mercado mas ainda é utilizado sobre hardware fechado;
- Hardware proprietário *não é ruim*, porém em suas características;
- O hardware proprietário nos serviu bem então por que precisamos de mudanças?



Por que devo me preocupar?

- **Lei de Moore (1965):** o número de transistores em um processador irá dobrar a cada ano;
- Limitações físicas de tamanho/consumo de energia está fazendo a indústria repensar;
- A abertura de especificações traz inovações e visões de fora incentivando a inovação;

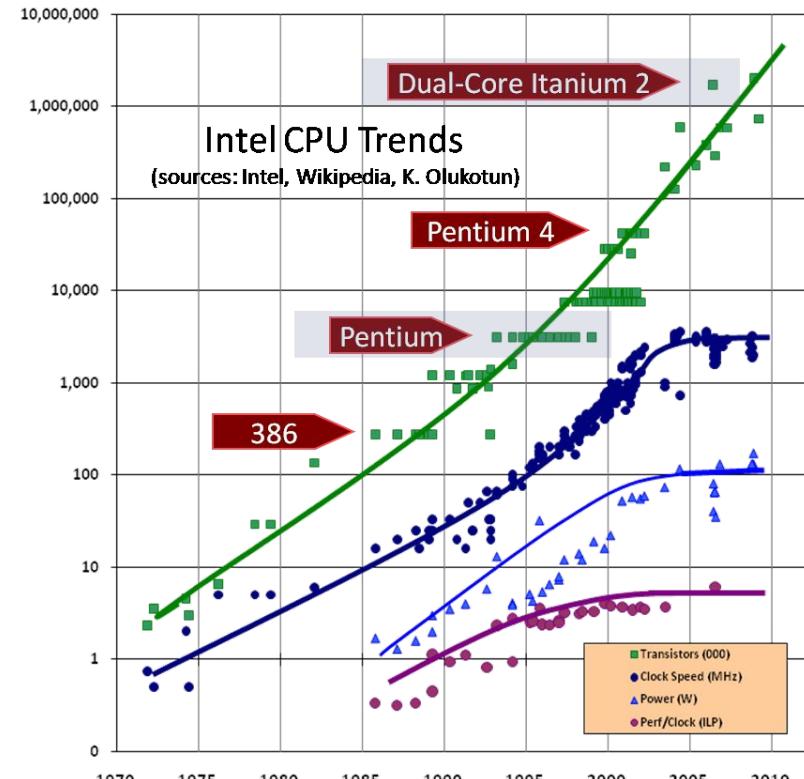
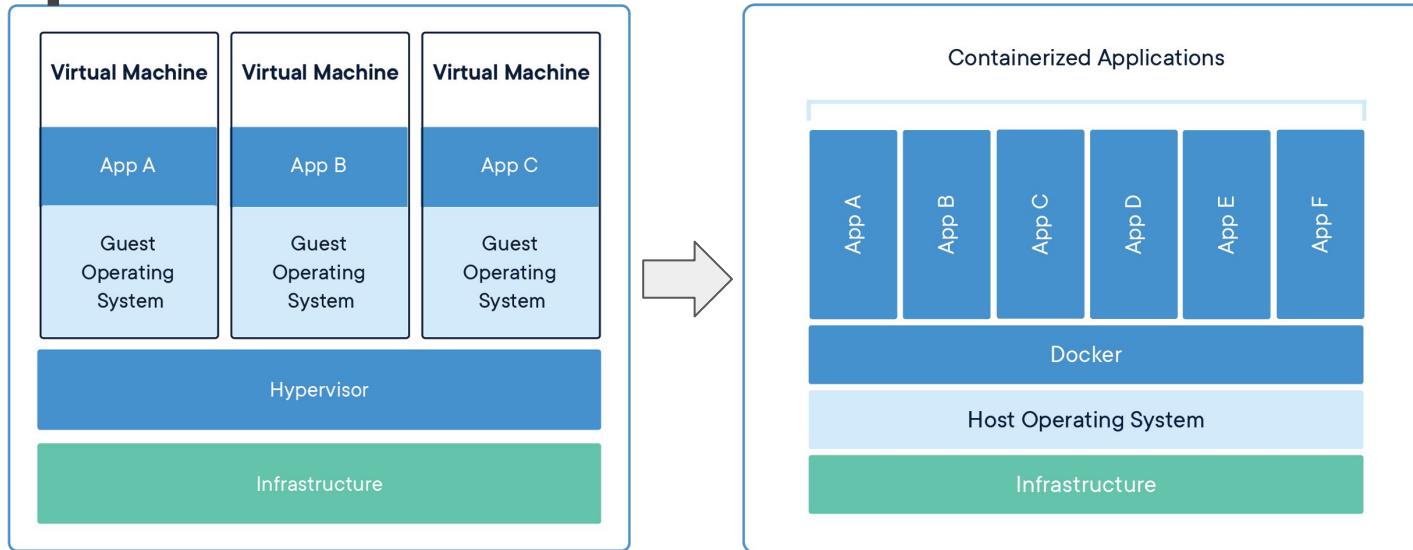


Image Credit: *The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software*,
Herb Sutter

O que são containers?



- Uma forma padronizada de execução de software
- Empacotar todos os requisitos da aplicação em uma imagem
- Pode ser executada em qualquer lugar, desde o laptop do desenvolvedor até servidores em um Data Center ou uma Cloud pública sem mudanças.

Containers on RISC-V



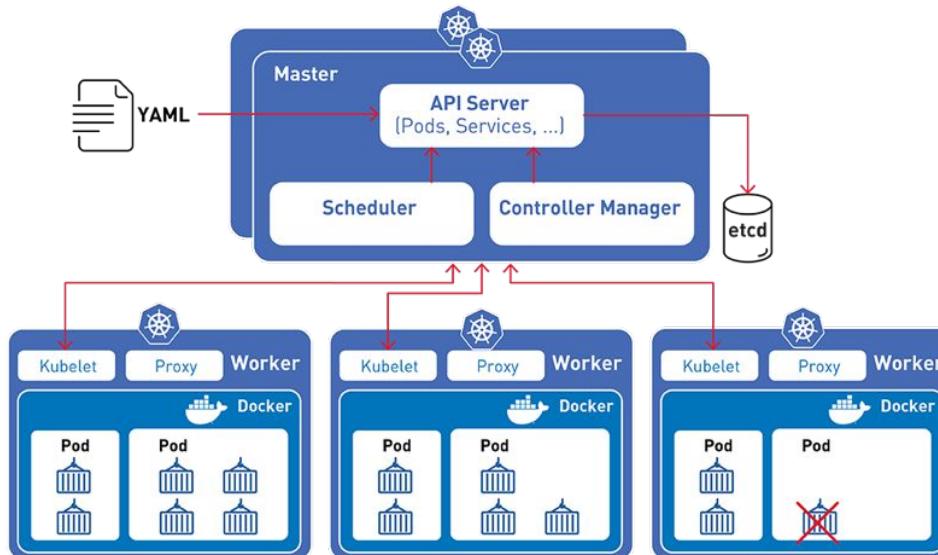
```
○ > 🐳 ~/echo
> uname -a
Linux qemuriscv 5.3.0-rc4-g6c2a8bd-dirty #6 SMP Fri Nov 22 18:08:43 -02 2019 riscv64 GNU/Linux

○ > 🐳 ~/echo
> DOCKER_BUILDKIT=1 docker build -t carlosedp/echo-riscv .
[+] Building 0.3s (5/5) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 37B
=> [internal] load build context
=> => transferring context: 34B
=> CACHED [1/1] COPY echo-riscv /echo-riscv
=> exporting to image
=> => exporting layers
=> => writing image sha256:231b741f511088f6f3901b8877ba8130f8e14a0780fa5ae261dd1f1c4d6b81f2
=> => naming to docker.io/carlosedp/echo-riscv

○ > 🐳 ~/echo
> docker run --name hello -d -p 8085:8080 carlosedp/echo-riscv
bf363e2f2b19a7171c88664e685289882a167d00c3118bfe77138ef11aae9899

○ > 🐳 ~/echo
> curl localhost:8085
Hello, I'm running Echo inside a container on linux/riscv64%
```

And Container Orchestration?



Kubernetes em RISC-V



Diversos Pull Requests depois

Project	Description	Pull Request	Upstream
go/net	ipv4, ipv6, internal/socket: add riscv64 support	https://github.com/golang/net/pull/43	✓
go/sys	unix: add riscv64 tag to endian_little.go	https://github.com/golang/sys/pull/10	✓
mattn/go-isatty	Update x/sys to support Risc-V	https://github.com/mattn/go-isatty/pull/1	✓
creack/pty	Add riscv64 support	https://github.com/creack/pty/pull/1	✓
docker/cli	Add riscv64 to manifest annotation and bash completion	https://github.com/docker/cli/pull/100	✓
opencontainers/runc	Bump x/sys and update syscall for initial Risc-V support	https://github.com/opencontainers/runc/pull/100	✓
moby/moby	bump x/sys to fix riscv64 epoll	https://github.com/moby/moby/pull/389	
moby/moby	Update modules to support riscv64	https://github.com/moby/moby/pull/400	
moby/moby	allow dockerd builds without cgo - Tonis	https://github.com/moby/moby/pull/401	
containerd/containerd	bump x/sys to fix riscv64 epoll	https://github.com/containerd/containerd/pull/100	
containerd/containerd	Update x/sys, x/net and bbolt modules to support Risc-V architecture	https://github.com/containerd/containerd/pull/101	
docker/cli	bump x/sys to fix riscv64 epoll	https://github.com/docker/cli/pull/102	✓
docker/libnetwork	bridge: add riscv64 build tags - Tonis	https://github.com/docker/libnetwork/pull/2389	✓
LK4D4/vndr	add riscv64 support	https://github.com/LK4D4/vndr/pull/80	✓
vishvananda/netns	add riscv64 architecture	https://github.com/vishvananda/netns/pull/34	✓
containers/libpod	build: allow to build without cgo on RISC-V	https://github.com/containers/libpod/pull/3437	✓

20+ Projetos
40+ PRs
6 CNCF Projetos
95%+ Upstream

Imagenes de containers para RISC-V

OpenFaaS:

- faas-gateway - `carlosedp/faas-gateway:riscv64`
- faas-basic-auth-plugin - `carlosedp/faas-basic-auth-plugin:riscv64`
- faas-swarm - `carlosedp/faas-swarm:riscv64`
- faas-netes - `carlosedp/faas-netes:riscv64`
- nats-streaming - `carlosedp/faas-nats-streaming:riscv64`
- queue-worker - `carlosedp/faas-queue-worker:riscv64`
- watchdog - `carlosedp/faas-watchdog:riscv64`
- Function base - `carlosedp/faas-debianfunction:riscv64`
- Figlet - `carlosedp/faas-figlet:riscv64`
- MarkdownRender - `carlosedp/faas-markdownrender:riscv64`
- QRCode - `carlosedp/faas-qrcode:riscv64`

Prometheus:

- Prometheus - `carlosedp/prometheus:v2.11.1-riscv64`
- AlertManager - `carlosedp/alertmanager:v0.18.0-riscv64`

Traefik:

- traefik v2 - `carlosedp/traefik:v2.1-riscv64`
- whoami - `carlosedp/whoami:riscv64`

Kubernetes:

- kube-apiserver - `carlosedp/kube-apiserver:1.16.0`
- kube-scheduler - `carlosedp/kube-scheduler:1.16.0`
- kube-controller-manager - `carlosedp/kube-controller-manager:1.16.0`
- kube-proxy - `carlosedp/kube-proxy:1.16.0`
- pause - `carlosedp/pause:3.1`
- flannel - `carlosedp/flanneld:v0.11.0`
- etcd (v3.5.0) - `carlosedp/etcd:3.3.10`
- CoreDNS (v1.6.3) - `carlosedp/coredns:v1.6.2`

Kubernetes images are multi-arch with manifests to `arm`, `arm64`, `amd64`, `riscv64` and `ppc64le`. Some version mismatches due to Kubernetes hard-coded version check for CoreDNS and etcd.

Misc Images:

- Echo demo - `carlosedp/echo-riscv`
- Whoami (Traefik demo) - `carlosedp/whoami:riscv64`
- Kubernetes Pause - `carlosedp/k8s-pause:riscv64`
- CoreDNS v1.3.0 - `carlosedp/coredns:v1.3.0-riscv64`

Ainda há muito a alcançar

Project	Description	Status
Golang	Go programming language	<input checked="" type="checkbox"/> Already upstream in 1.15. Pending binary distribution
SECCOMP	Risc-V Seccomp support to Linux Kernel	<input checked="" type="checkbox"/> Upstream from Kernel 5.5
Libseccomp	Upstream libseccomp patch	<input checked="" type="checkbox"/> Already upstream
Kubernetes	Have Kubernetes build on Risc-V	<input checked="" type="checkbox"/> Builds on RISC-V. Still not officially supported.
Golang CGO	Go extension for importing C code	<input checked="" type="checkbox"/> Already merged. Available on Go 1.16 (2021)
Runc	Container Runtime in C	<input checked="" type="checkbox"/> Already can be built with Go 1.16.
Golang SQLite	File database libraries	<input checked="" type="checkbox"/> Already can be built with Go 1.16.
Official Distros	Official Linux distributions	Still pending official support from most distros (Fedora, Ubuntu, Debian, Alpine)
Official Images	Build and publish official images	Depends on distro support (Debian, Alpine, Fedora)
Most projects	Build and distribute binaries and/or images	Changes on CI/CD and official images

Demo





Obrigado!

Twitter:

<https://twitter.com/carlosedp>

Artigos:

<https://carlosedp.com>

Acompanhe o progresso em:

<https://github.com/carlosedp/riscv-bringup>

<https://hub.docker.com/u/carlosedp>