



arm

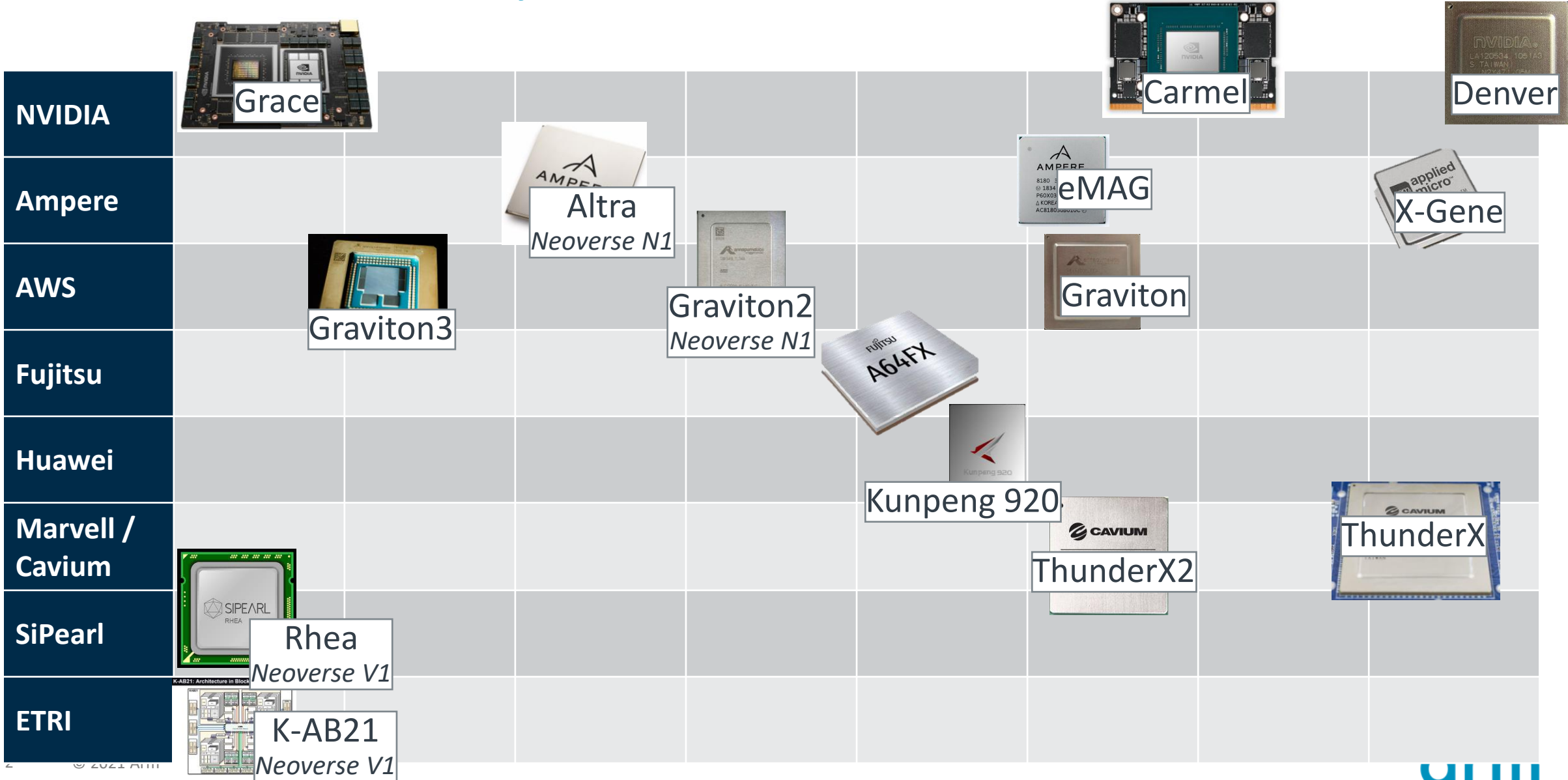
How to Optimize for Arm and not get Eaten by a Bear

*Performance Optimization in a
World of Multiple Microarchitectures*

john.linford@arm.com

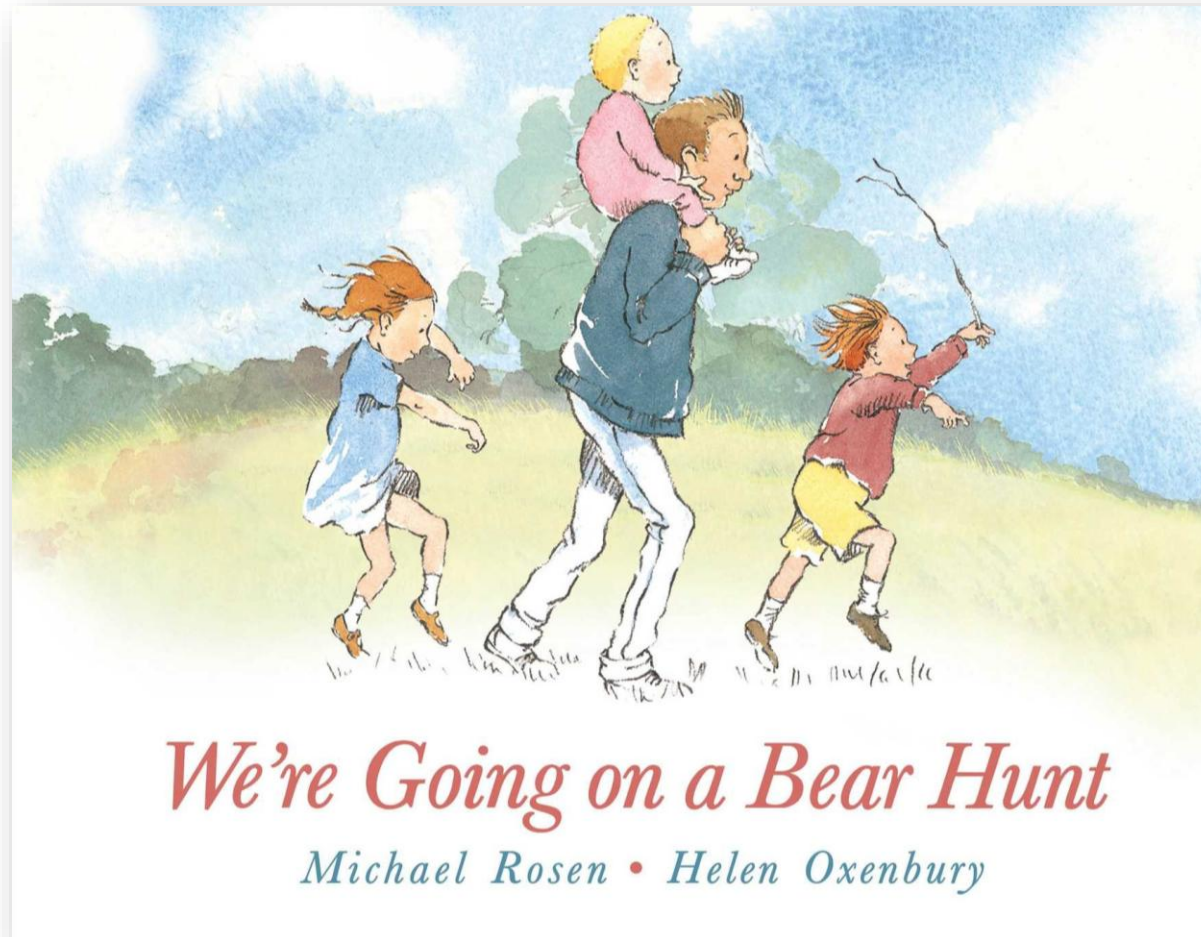
10 Feb 2022

Arm Enables Diversity



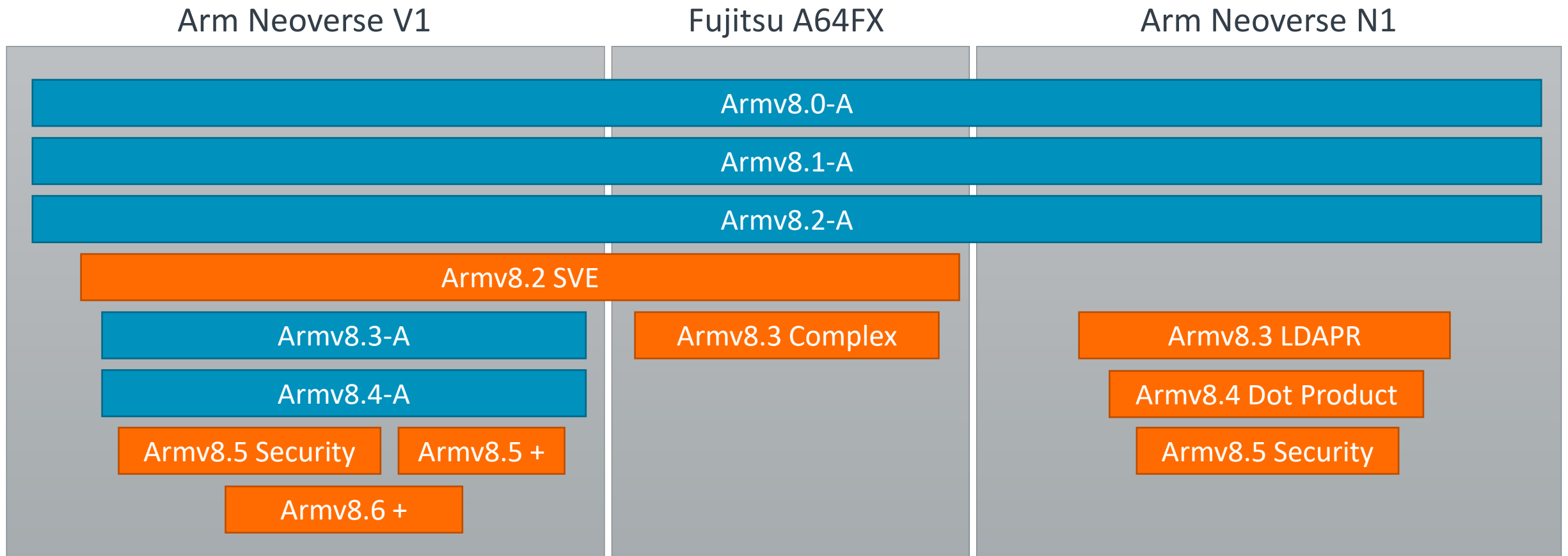
How to hunt the bear and not get eaten?

How to “optimize for Arm” without becoming tied to a specific chip?

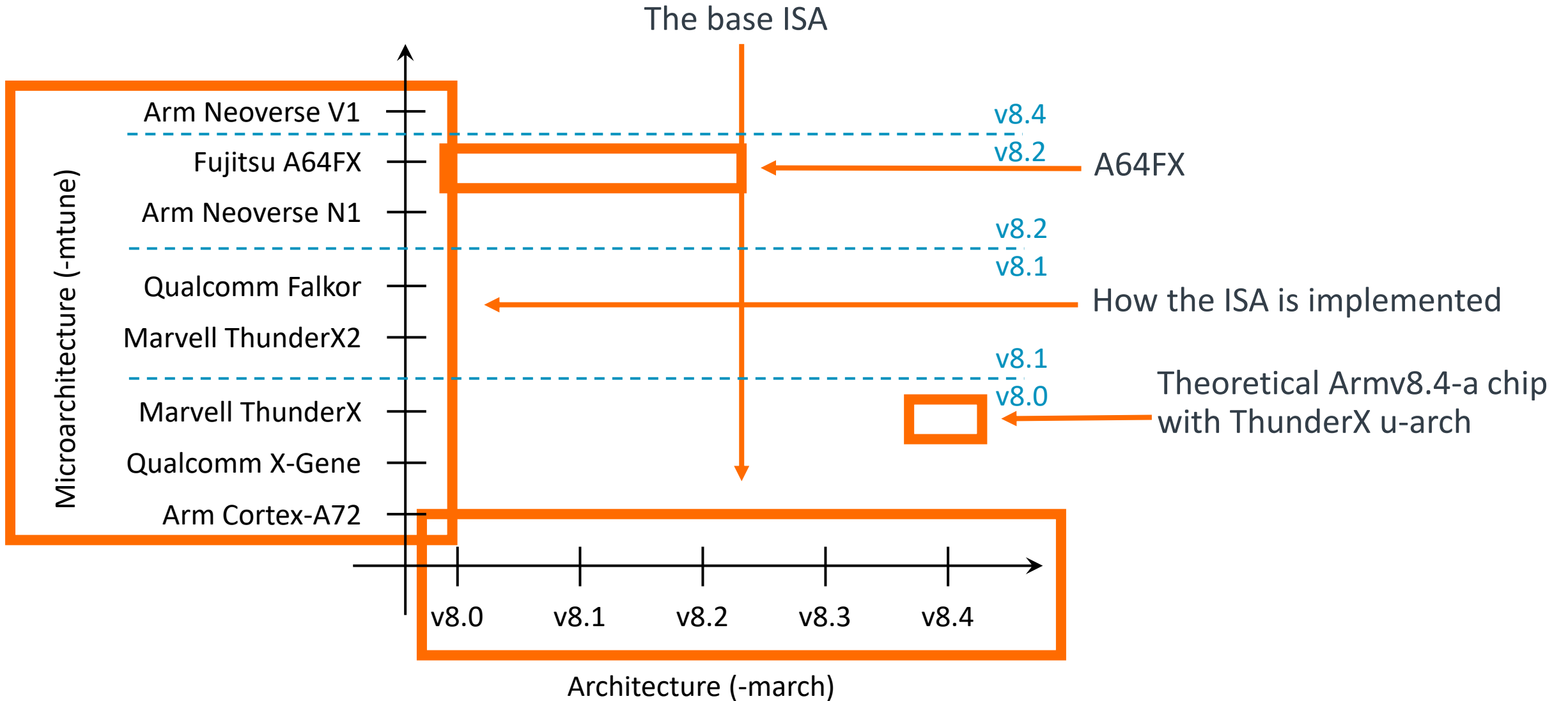


Core Instruction Set Architecture (ISA)

A CPU's vocabulary



ISA vs. u-arch



How to specify ISA and u-arch?

GCC and LLVM (and LLVM-based compilers)

-march

- For aarch64 targets, this flag specifies the ISA
- Fine-grained control over ISA extensions: “armv8.2-a+sve”
- This flag behaves differently for x86 targets!

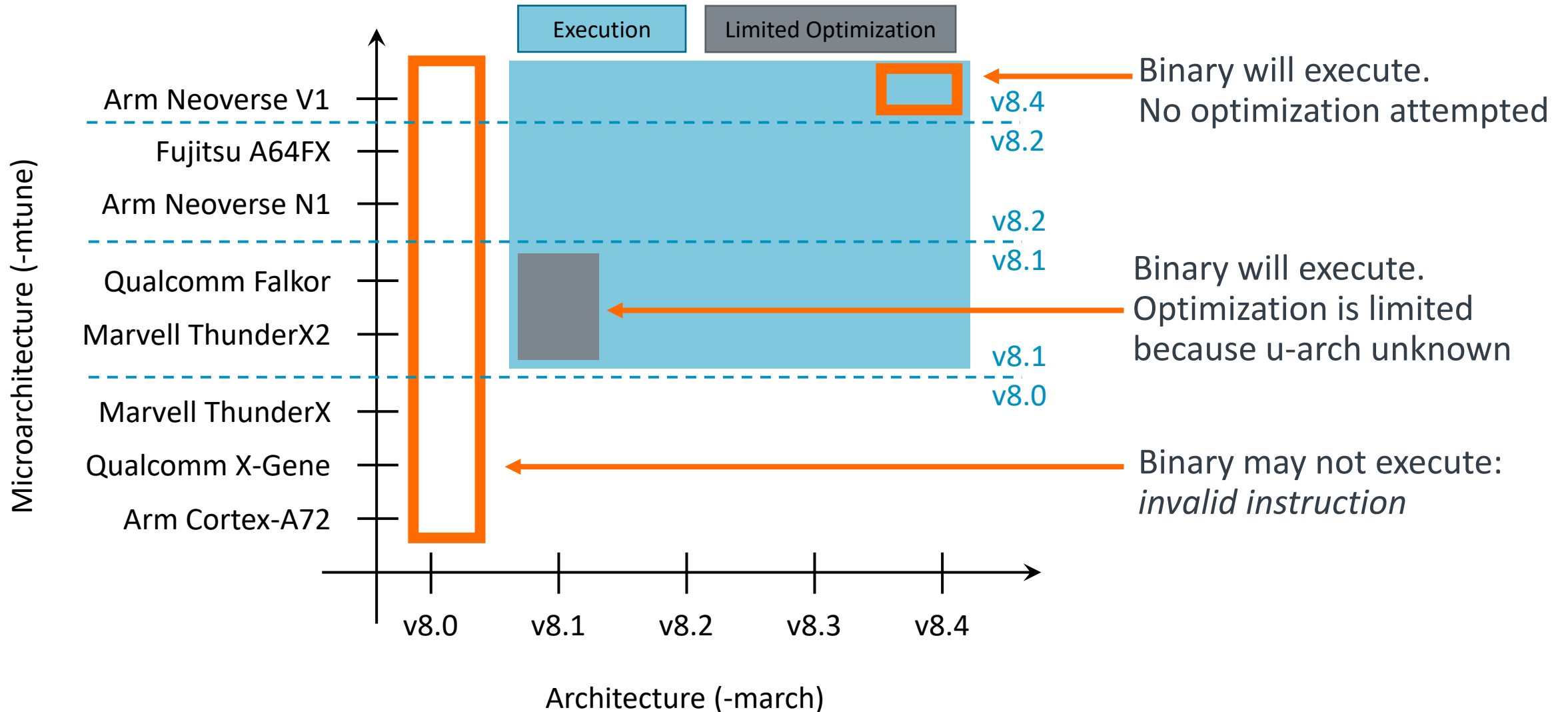
-mtune

- For aarch64 targets, this flag specifies the u-arch
- See man pages for supported u-arches
- This flag behaves differently for x86 targets!

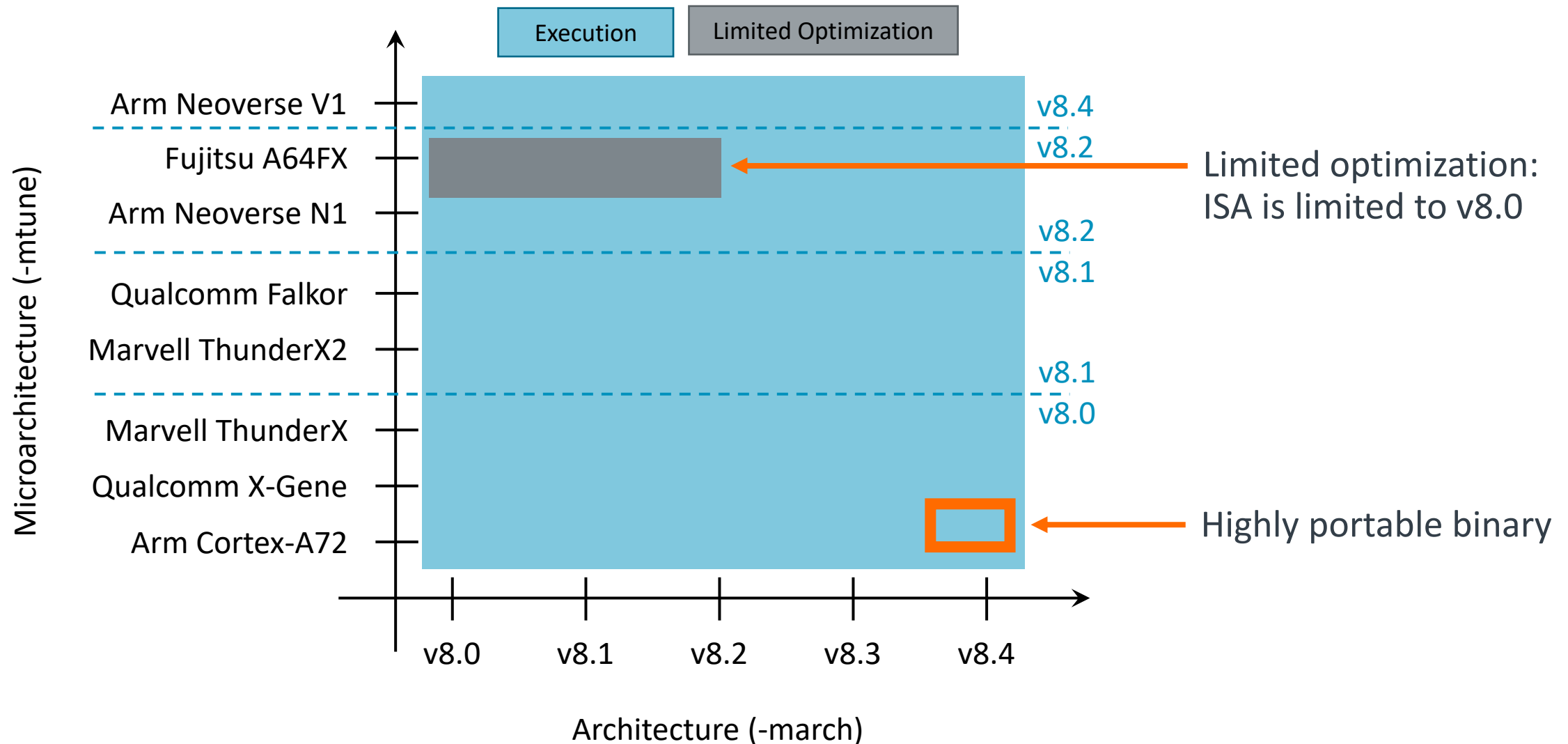
-mcpu

- For aarch64 targets, this flag is a shortcut. It specifies both the ISA and the u-arch
- Accepts the same parameters as -mtune
- This flag is deprecated for x86 targets!

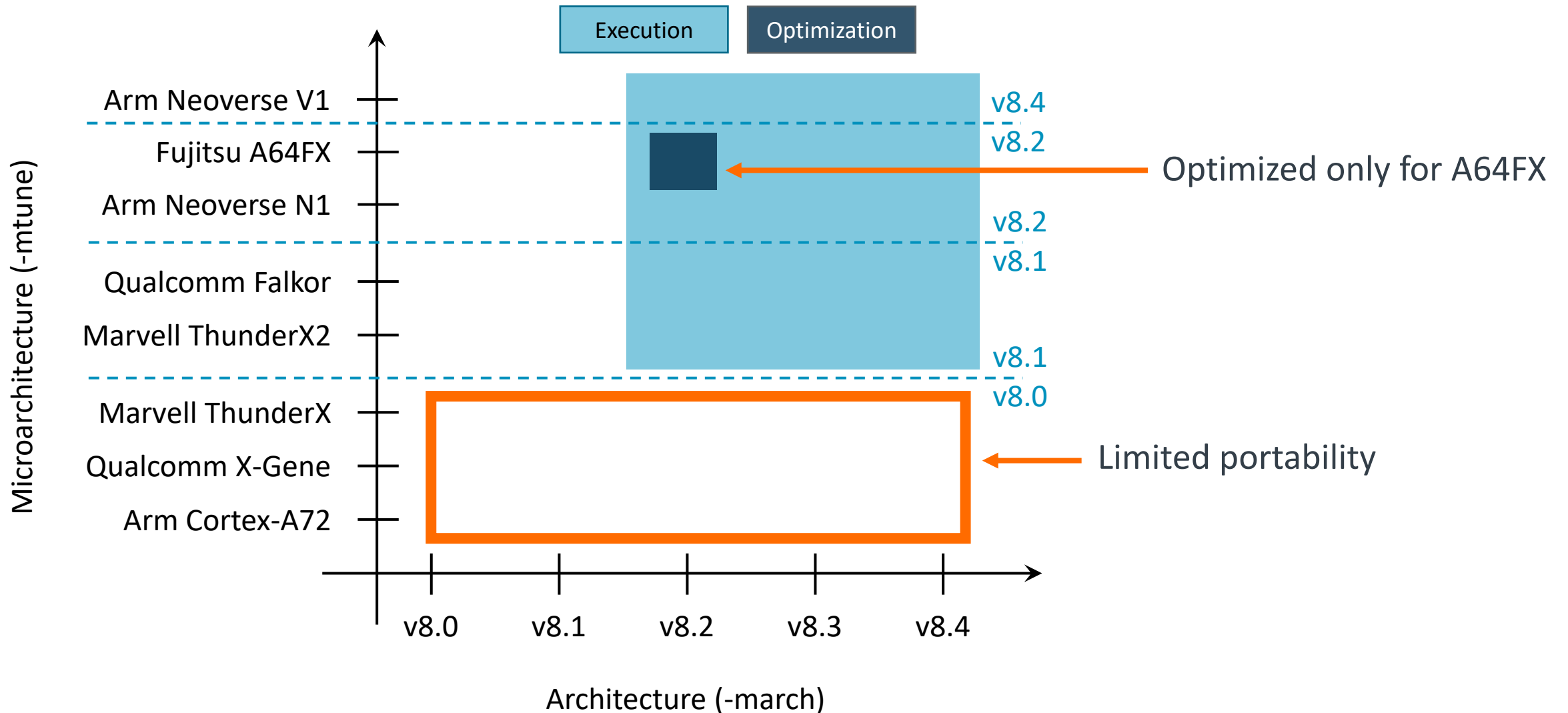
Execution vs. Optimization: `-march=armv8.1`



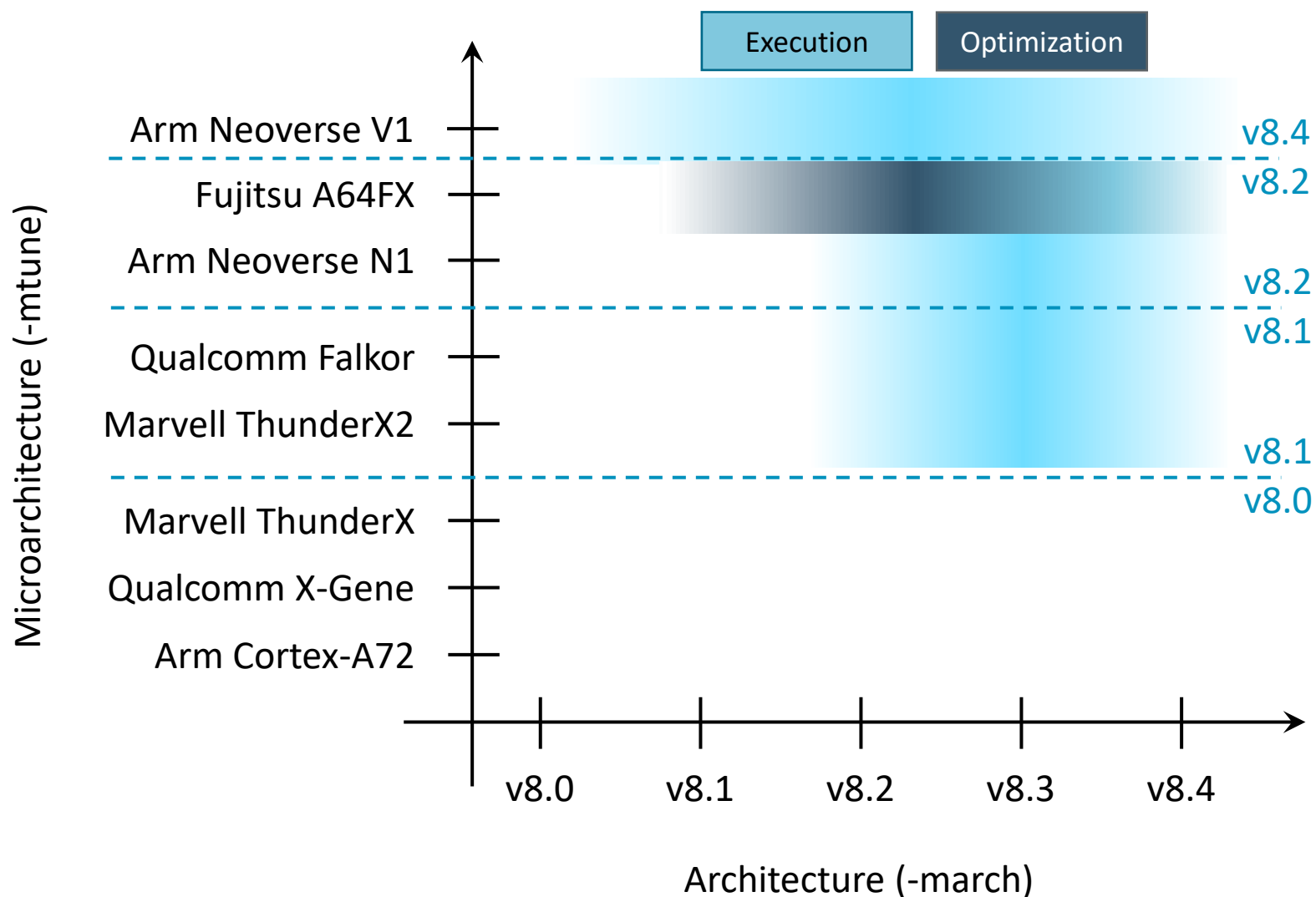
Execution vs. Optimization: `-mtune=a64fx`



Execution vs. Optimization: `-mcpu=a64fx`



A more realistic view of `-mcpu=a64fx`



Binary is expected to *only* be used on A64FX.
Compiler is free to use extensions, take shortcuts, etc.

__sync_fetch_and_add(&var, num);

GCC 11.1.0 on A64FX

-march=armv8.2-a

No SVE

```
.arch armv8.2-a+crc
.file "foo.c"
.text
.section .rodata
.align 3
```

```
mov x29, sp
str w0, [sp, 28]
str x1, [sp, 16]
ldr w0, [sp, 28]
str w0, [sp, 44]
mov w0, 1
str w0, [sp, 40]
ldr w0, [sp, 40]
mov w1, w0
add x0, sp, 44
ldaddal w1, w0, [x0]
ldr w0, [sp, 44]
mov w1, w0
adrp x0, .LC0
add x0, x0, :lo12:.LC0
bl printf
mov w0, 0
ldp x29, x30, [sp], 48
```

Atomic
Add

-mtune=a64fx

Minimal ISA

```
.arch armv8-a
.file "foo.c"
.text
.global __aarch64_ldadd4_acq_rel
.section .rodata
.align 3
```

```
mov x29, sp
str w0, [sp, 28]
str x1, [sp, 16]
ldr w0, [sp, 28]
str w0, [sp, 44]
mov w0, 1
str w0, [sp, 40]
ldr w0, [sp, 40]
mov w2, w0
add x0, sp, 44
mov x1, x0
mov w0, w2
bl __aarch64_ldadd4_acq_rel
ldr w0, [sp, 44]
mov w1, w0
adrp x0, .LC0
add x0, x0, :lo12:.LC0
bl printf
mov w0, 0
ldp x29, x30, [sp], 48
```

libgcc call

-mcpu=a64fx

Best ISA

```
.arch armv8.2-a+crc+sve
.file "foo.c"
.text
.section .rodata
.align 3
```

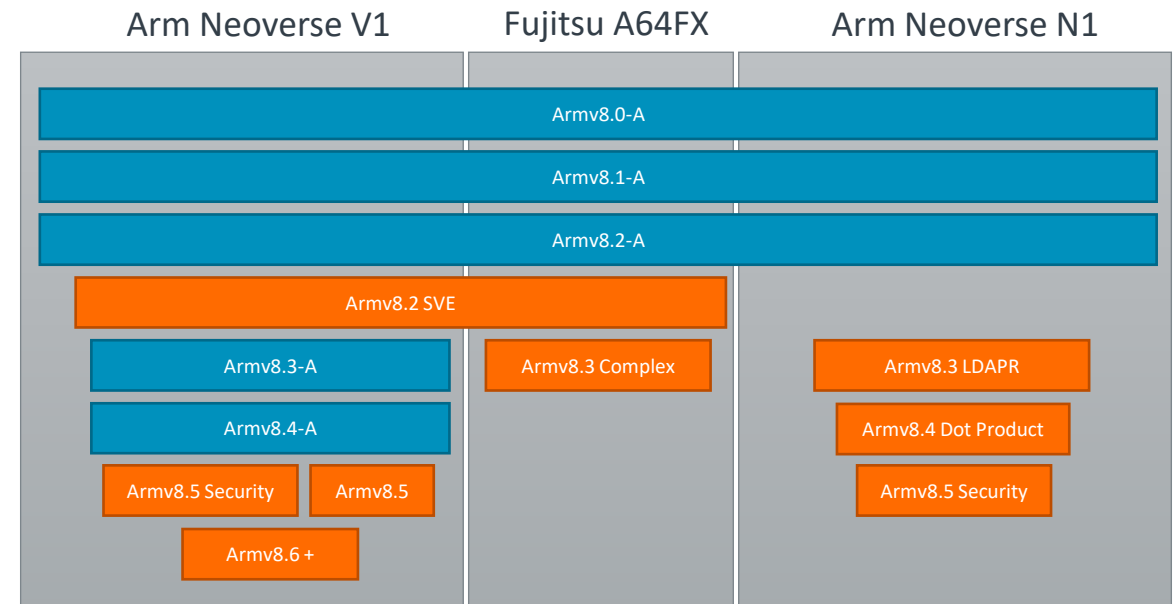
```
mov x29, sp
str w0, [sp, 28]
str x1, [sp, 16]
ldr w0, [sp, 28]
str w0, [sp, 44]
mov w0, 1
str w0, [sp, 40]
ldr w0, [sp, 40]
mov w1, w0
add x0, sp, 44
ldaddal w1, w0, [x0]
ldr w0, [sp, 44]
mov w1, w0
adrp x0, .LC0
add x0, x0, :lo12:.LC0
bl printf
mov w0, 0
ldp x29, x30, [sp], 48
```

Atomic
Add

Compiler flags for tuned portable binaries

<https://gcc.gnu.org/onlinedocs/gcc/AArch64-Options.html#aarch64-feature-modifiers>

- V1-optimized, runs on A64FX
 - `-march=armv8.2-a+sve`
 - `-mtune=neoverse-v1`
 - `-msve-vector-bits=scalable`
 - Targets V1 u-arch and limits the ISA to A64FX
 - Uses both SVE and NEON, and will occasionally prefer NEON over SVE
- V1-optimized, runs on N1
 - `-march=armv8.2-a+nosve+dotprod`
 - `-mtune=neoverse-v1`
 - Targets V1 u-arch and limits the ISA to N1
 - Uses only NEON (which performs well on V1)
- N1-optimized, runs on V1
 - `-mcpu=neoverse-n1`
 - V1's features are a superset of N1's



How to hunt the bear and not get eaten

a.k.a how to get good performance without being tied to a particular chip

- Let someone else hunt the bear
 - NVIDIA NGC
 - wiki.arm-hpc.org
- Only hunt the bear where it is safe
 - Link against portable optimized libraries
 - Use autovectorizing compilers – don't hand-tune SIMD code
- If you must hunt the bear, stay outside the cave
 - Compile for a common base architecture
 - If extensions are critical to your code's performance, understand that cost
- If you enter the cave, be sure you can run out
 - Build from source with the appropriate flags (e.g. Spack or EasyBuild)
 - Distribute multiple binaries and dynamically load as appropriate

arm

Thank You

Danke

Gracias

谢谢

ありがとう

Asante

Merci

감사합니다

धन्यवाद

Kiitos

شكرًا

ধন্যবাদ

תודה

arm

The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks