WHAT IS SYCL?

LEARNING OBJECTIVES

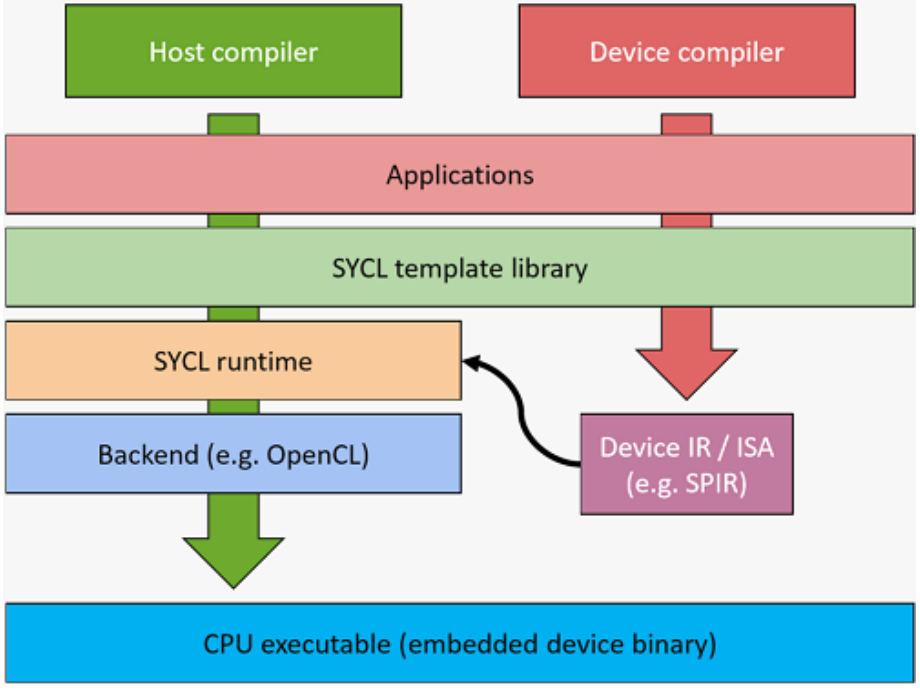
- Learn about the SYCL specification and it's implementations
- Learn about the components of a SYCL implementation
- Learn about how a SYCL source file is compiled
- Learn where to find useful resources for SYCL

WHAT IS SYCL?

SYCL is a single source, high-level, standard C++ programming model, that can target a range of heterogeneous platforms

WHAT IS SYCL?

SYCL is a **single source**, high-level, standard C++ programming model, that can target a range of heterogeneous platforms



- SYCL allows you to write both host CPU and device code in the same C++ source file
- This requires two compilation passes; one for the host code and one for the device code

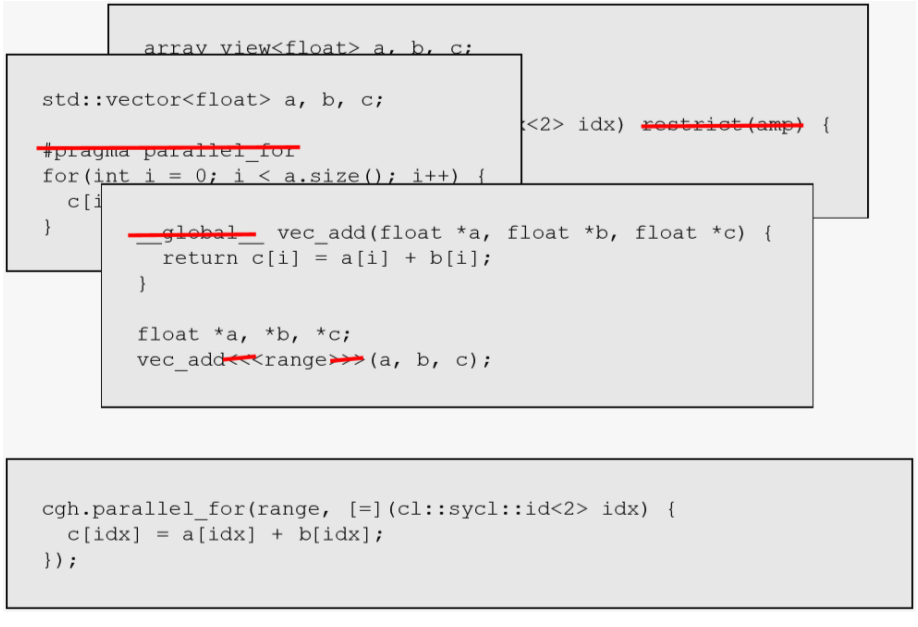
WHAT IS SYCL?

SYCL is a single source, **high-level**, standard C++ programming model, that can target a range of heterogeneous platforms

- SYCL provides high-level abstractions over common boilerplate code
  - Platform/device selection
  - Buffer creation and data movement
  - Kernel function compilation
  - Dependency management and scheduling

WHAT IS SYCL?

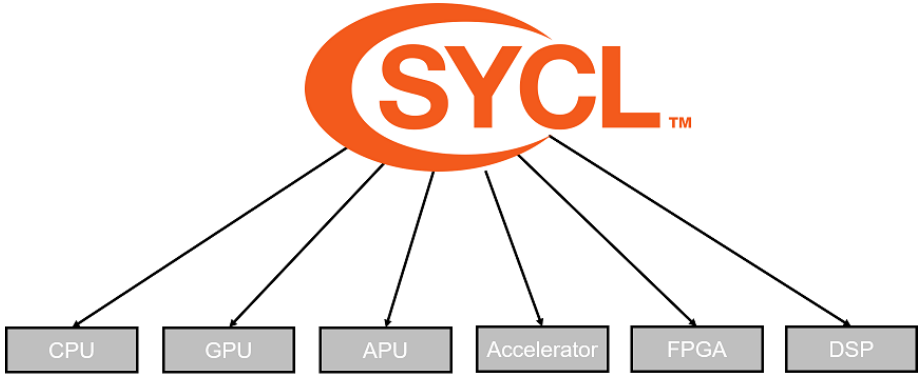
SYCL is a single source, high-level **standard C++** programming model, that can target a range of heterogeneous platforms



- SYCL allows you to write standard C++
  - SYCL 2020 is based on C++17
- Unlike the other implementations shown on the left there are:
  - No language extensions
  - No pragmas
  - No attributes

WHAT IS SYCL?

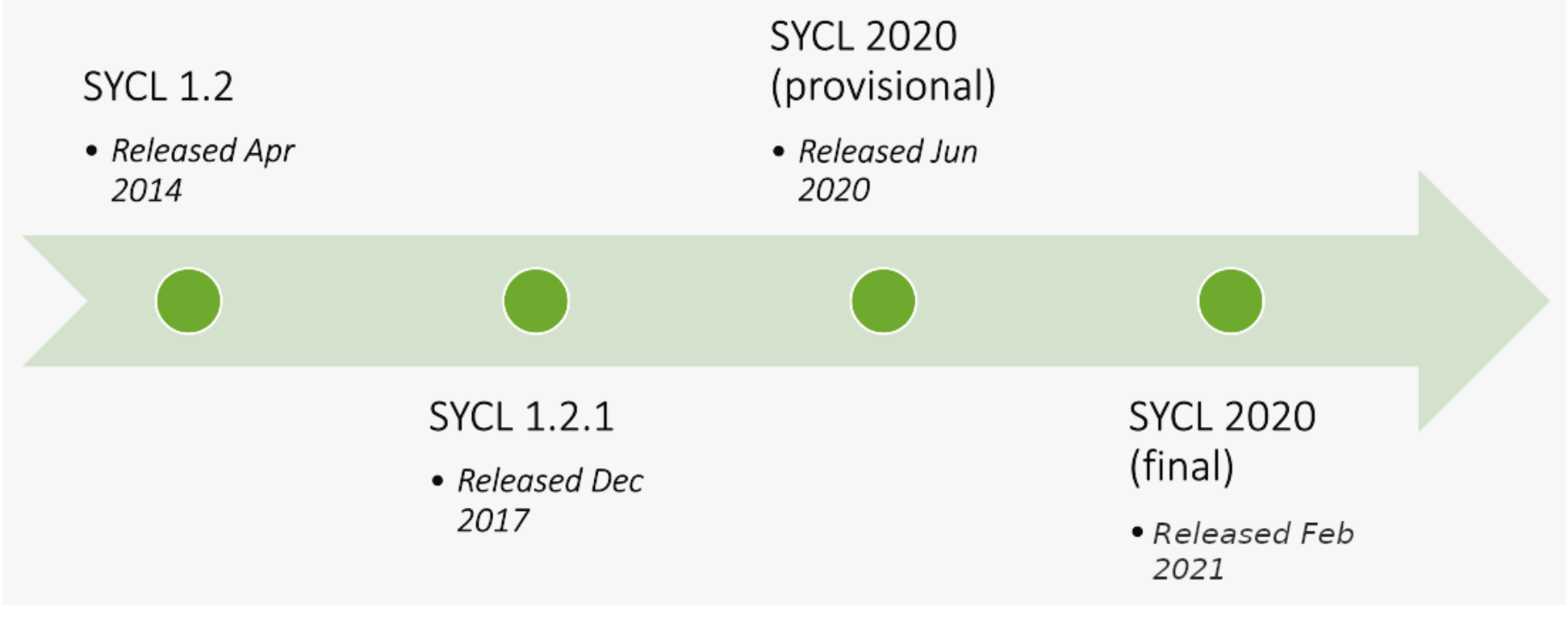
SYCL is a single source, high-level standard C++ programming model, that can **target a range of heterogeneous platforms**



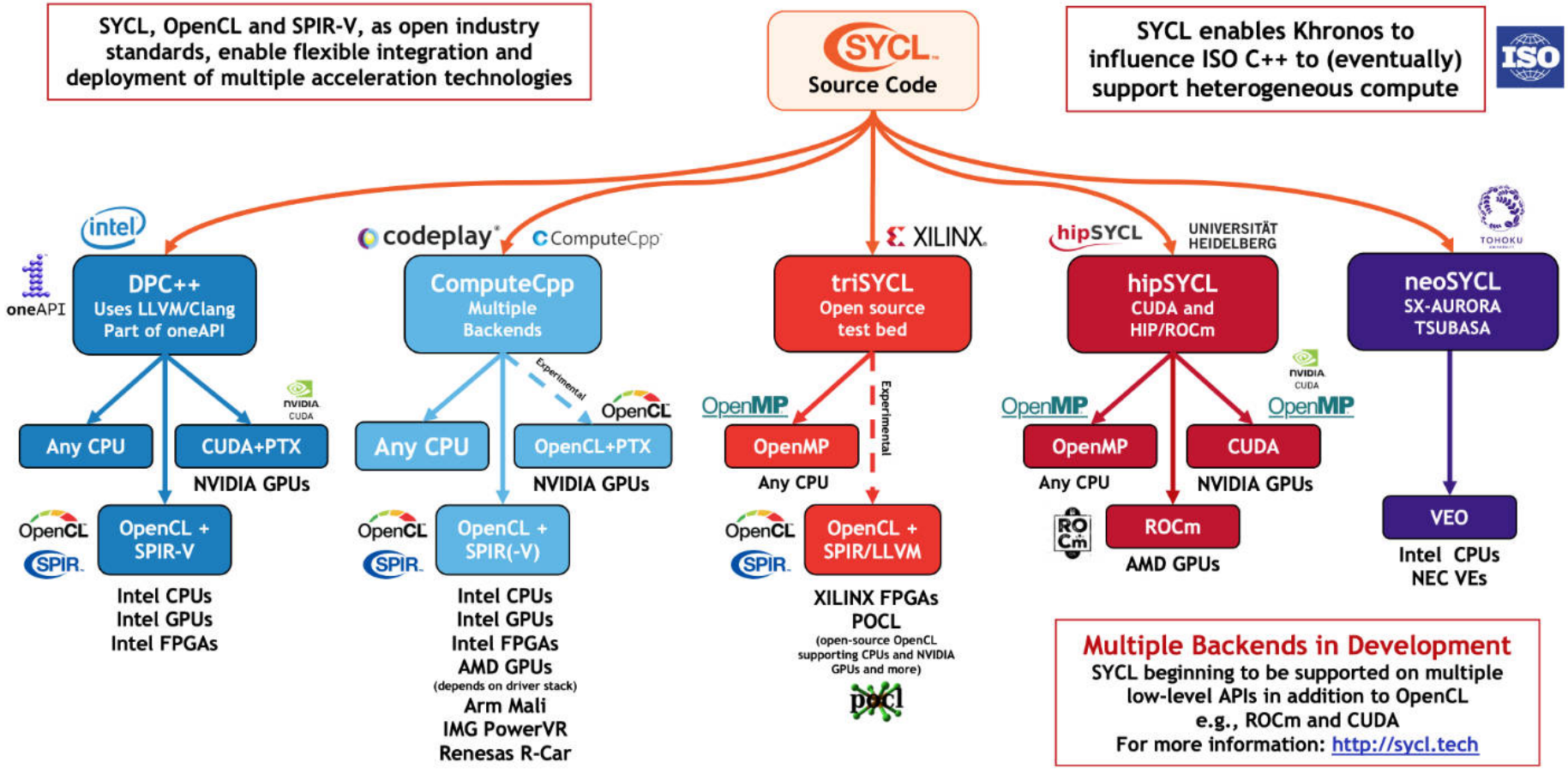
- SYCL can target any device supported by its backend
- SYCL can target a number of different backends

SYCL has been designed to be implemented on top of a variety of backends. Current implementations support backends such as OpenCL, CUDA, HIP, OpenMP and others.

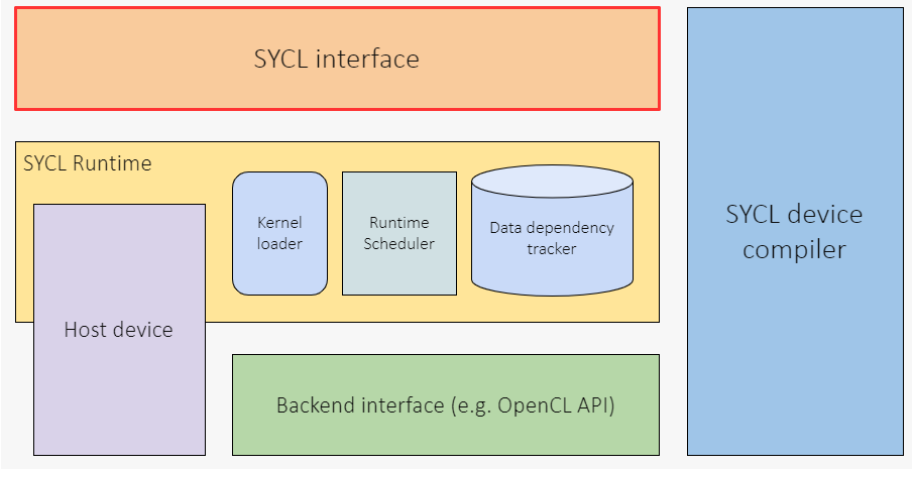
SYCL SPECIFICATION



SYCL IMPLEMENTATIONS



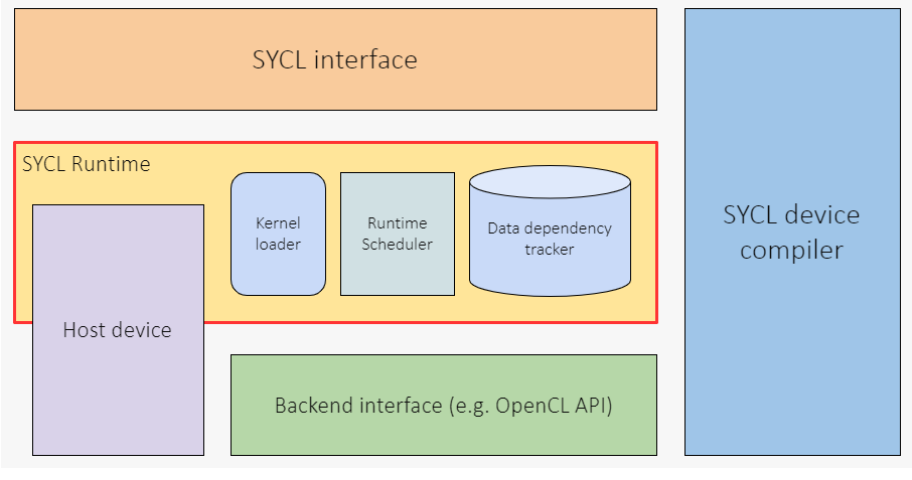
WHAT A SYCL IMPLEMENTATION LOOKS LIKE



- The SYCL interface is a C++ template library that developers can use to access the features of SYCL
- The same interface is used for both the host and device code

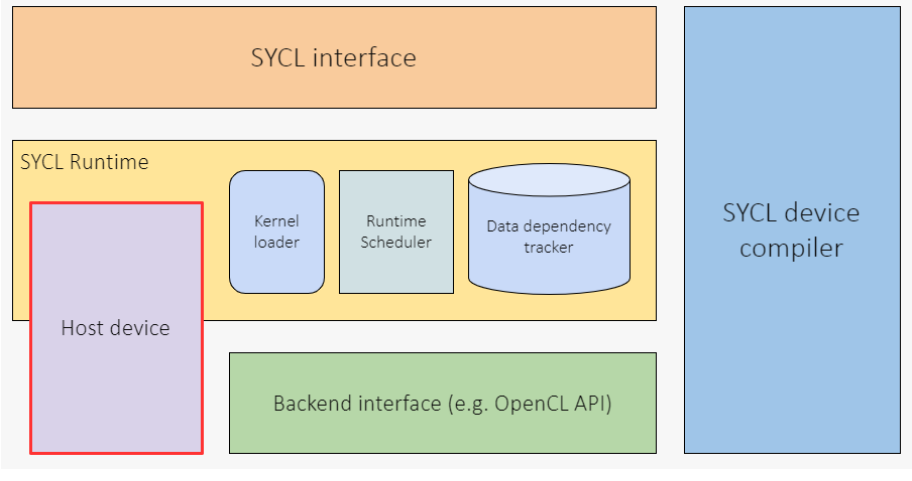
- The host is generally the CPU and is used to dispatch the parallel execution of kernels
- The device is the parallel unit used to execute the kernels, such as a GPU

WHAT A SYCL IMPLEMENTATION LOOKS LIKE



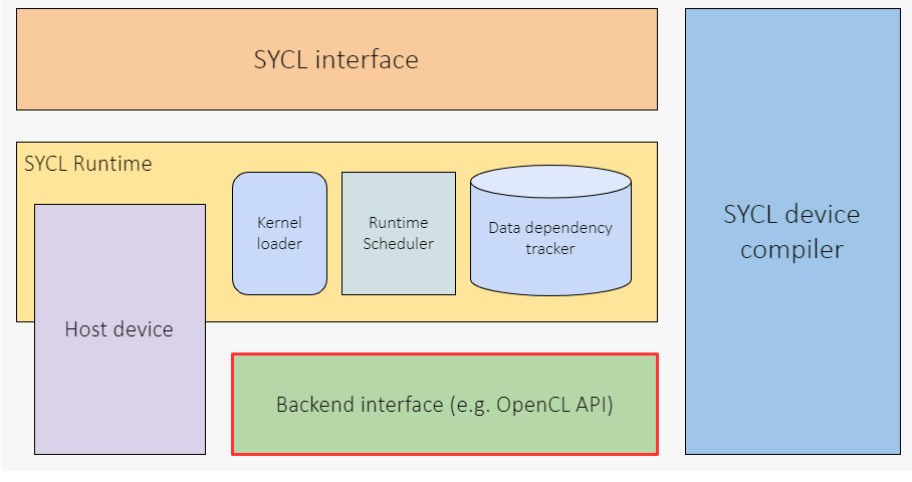
- The SYCL runtime is a library that schedules and executes work
  - It loads kernels, tracks data dependencies and schedules commands

WHAT A SYCL IMPLEMENTATION LOOKS LIKE



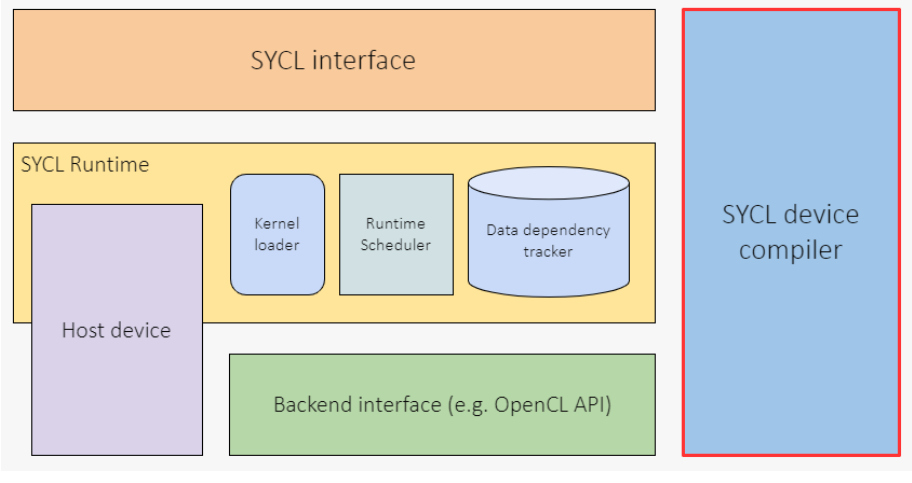
- The host device is an emulated backend that is executed as native C++ code and emulates the SYCL execution and memory model
- The host device can be used to execute kernels without backend drivers and for debugging purposes

WHAT A SYCL IMPLEMENTATION LOOKS LIKE



- The back-end interface is where the SYCL runtime calls down into a back-end in order to execute on a particular device
- Many implementations provide OpenCL backends, but some provide additional or different backends.

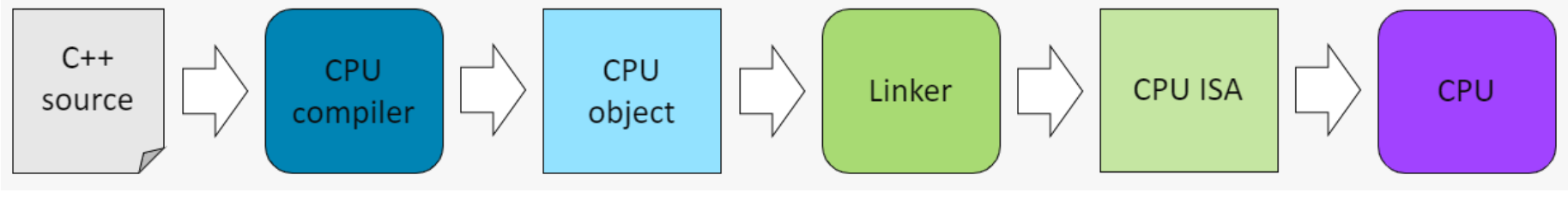
WHAT A SYCL IMPLEMENTATION LOOKS LIKE



- The SYCL device compiler is a C++ compiler which can identify SYCL kernels and compile them down to an IR or ISA
  - This can be SPIR, SPIR-V, GCN, PTX or any proprietary vendor ISA
- Some SYCL implementations are library only in which case they do not require a device compiler

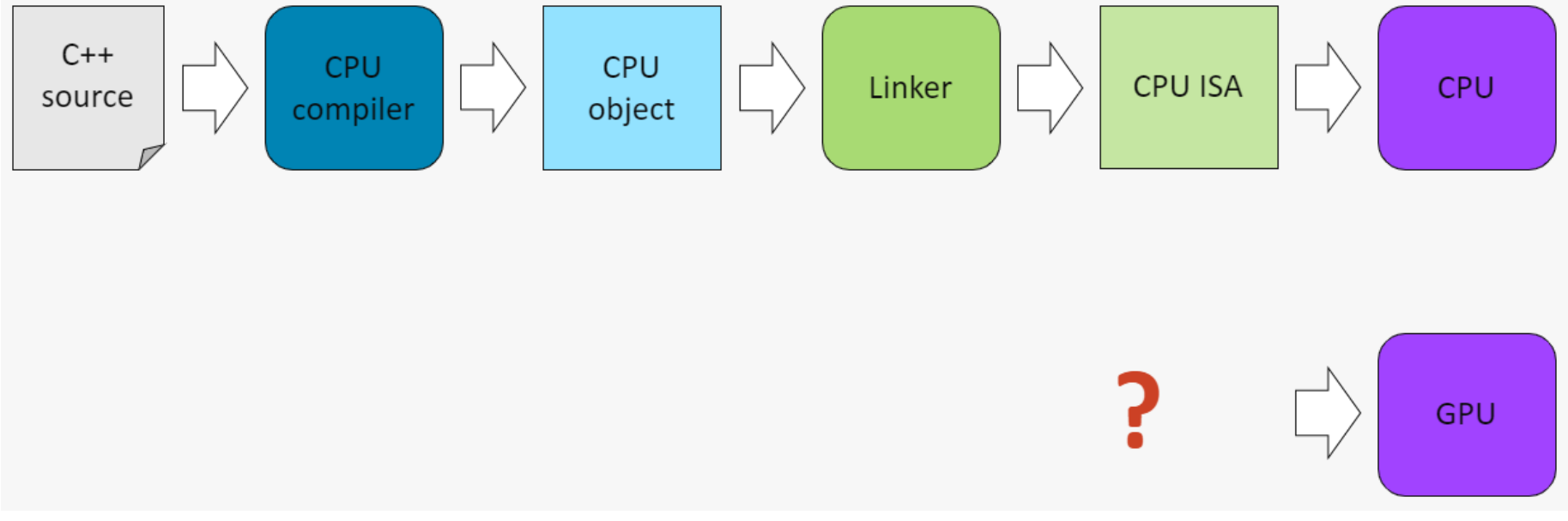
IR = Intermediate Representation **ISA** = Instruction Set Architecture

STD C++ COMPILATION MODEL



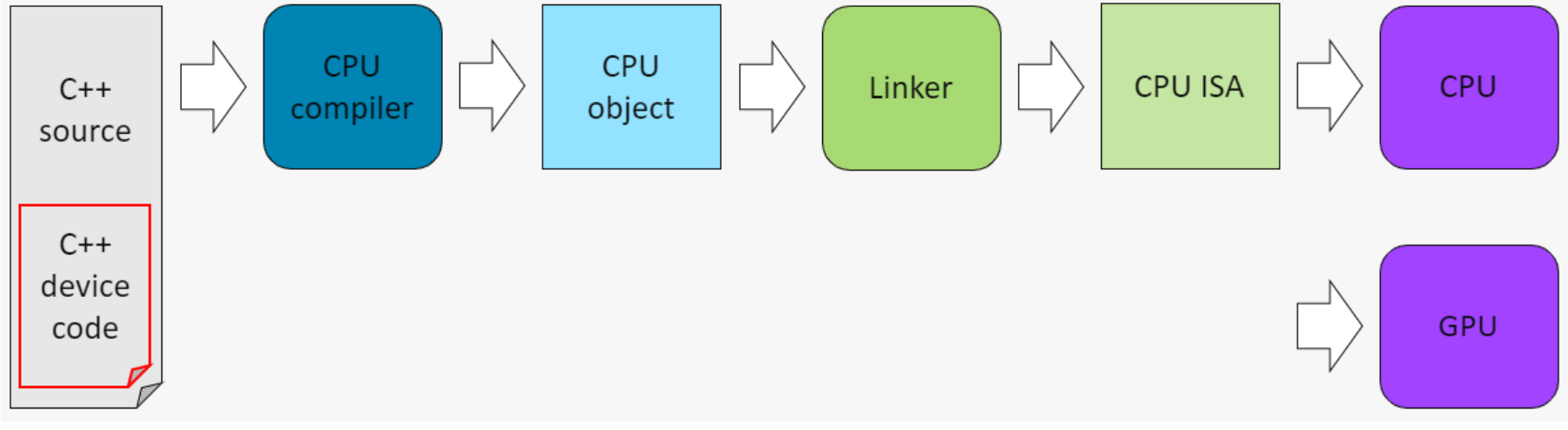
- This is the typical compilation model for a C++ source file.

STD C++ COMPILATION MODEL



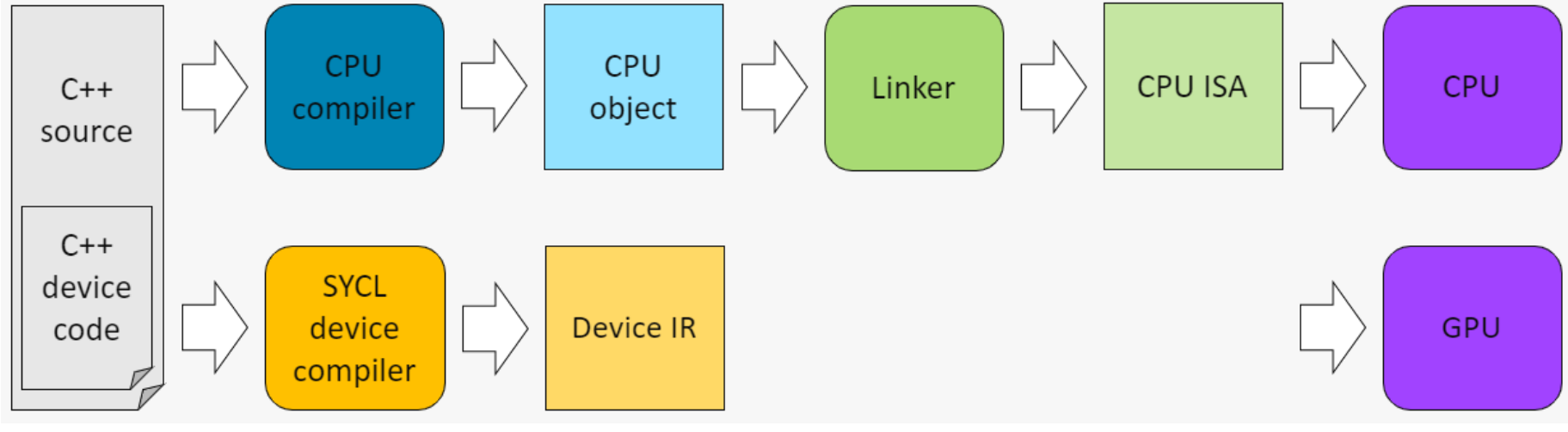
- So how do you compile a source file to also target the GPU?

STD C++ COMPILATION MODEL



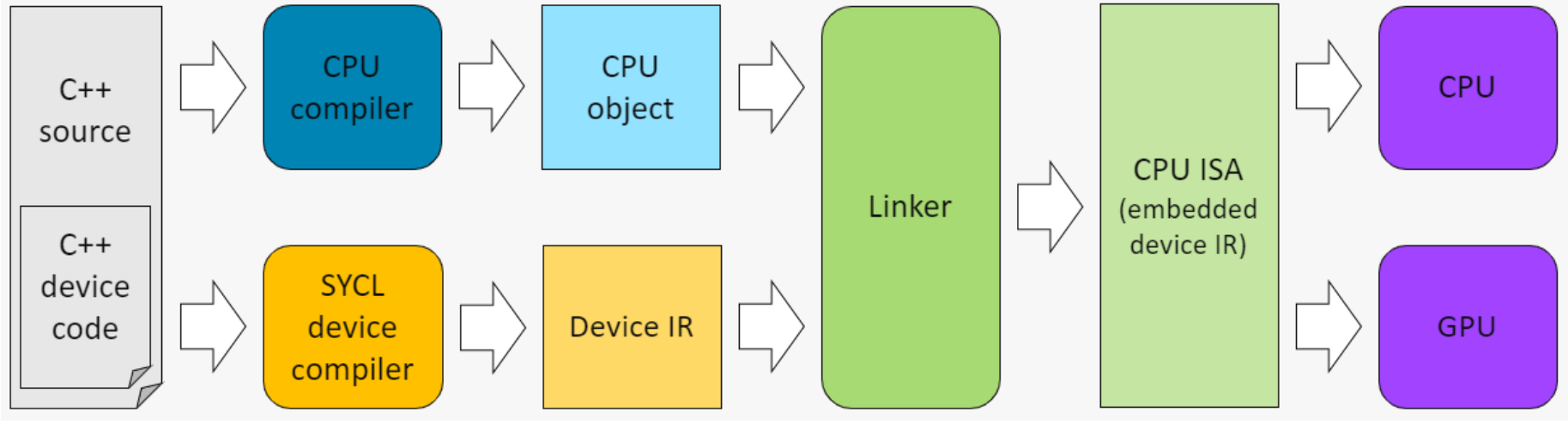
- As SYCL is single source the kernel functions are standard C++ function objects or lambda expressions.
- These are defined by submitting them to specific APIs.

STD C++ COMPILATION MODEL



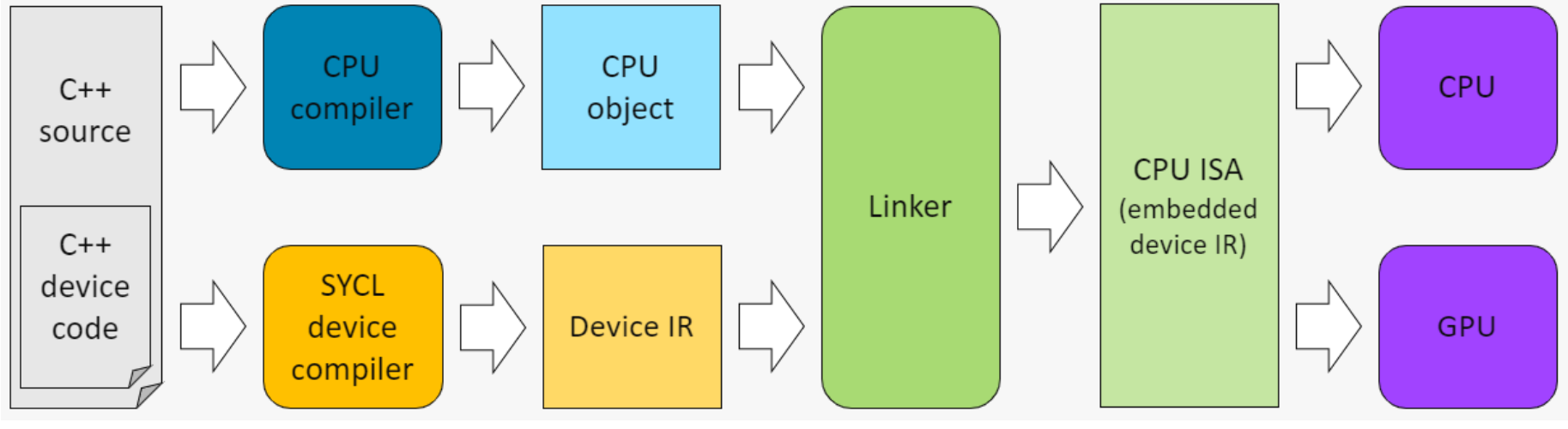
- As well as the standard C++ compiler, the source file is also compiled by a SYCL device compiler.
- This produces a device IR such as SPIR, SPIR-V or PTX or ISA for a specific architecture containing the GPU code.

STD C++ COMPILATION MODEL



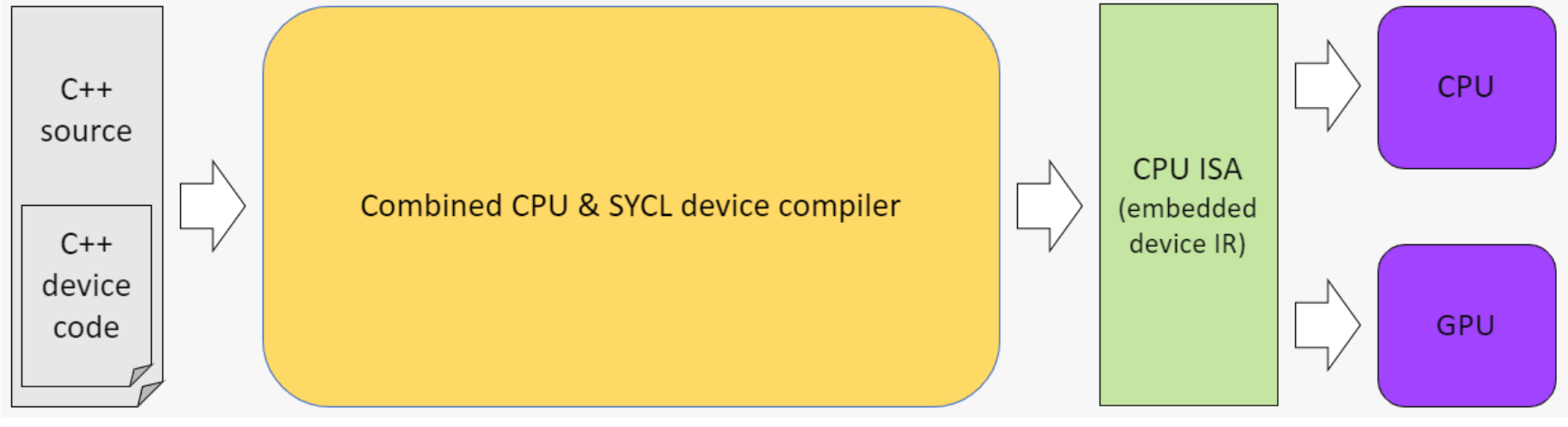
- The CPU object is then linked with the device IR or ISA to form a single executable with both the CPU and GPU code.

STD C++ COMPILATION MODEL



- This is the multi-compiler compilation model.
- This allows the host compiler (MSVC, Clang, Gcc) to be independent of the SYCL device compiler.

STD C++ COMPILATION MODEL



- SYCL also supports a single-compiler compilation model.
- Where both the host compiler and SYCL device compiler are invoked from the same driver.

WHERE TO GET STARTED WITH SYCL

- Visit https://www.khronos.org/sycl/ to find the latest SYCL specifications
- Checkout the documentation provided with one of the SYCL implementations.
- Visit https://sycl.tech to find out about all the SYCL implementations, news and videos

QUESTIONS

EXERCISE

Code_Exercises/Exercise_1_Compiling_with_SYCL/source

Configure your environment for using SYCL and compile a source file with the SYCL compiler.