Hands-On HPC Application Development Using C++ and SYCL

James Reinders, Phuong Nguyen, Thomas Applencourt, Rod Burns, Alastair Murray

WHAT IS SYCL?

LEARNING OBJECTIVES

- Learn about the SYCL specification and its 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?

A first example of SYCL code. Elements will be explained in coming sections!

```
1  #include <sycl/sycl.hpp>
2
3  int main(int argc, char *argv[]) {
4    std::vector<float> dA{2.3}, dB{3.2}, dO{7.9};
5
6    try {
7      auto asyncHandler = [&](sycl::exception_list eL) {
8        for (auto &e : eL)
9          std::rethrow_exception(e);
10     };
11     sycl::queue gpuQueue{sycl::default_selector{}, asyncHandler};
12
13     sycl::buffer bufA{dA.data(), sycl::range{dA.size()}};
14     sycl::buffer bufB{dB.data(), sycl::range{dB.size()}};
15     sycl::buffer bufO{dO.data(), sycl::range{dO.size()}};
16
17     gpuQueue.submit([&](sycl::handler &cgh) {
18       sycl::accessor inA(bufA, cgh, sycl::read_only);
19       sycl::accessor inB(bufB, cgh, sycl::read_only);
20       sycl::accessor out(bufO, cgh, sycl::write_only);
21
22       cgh.parallel_for(sycl::range{dA.size()},
23                        [=](sycl::id<1> i) { out[i] = inA[i] + inB[i]; });
24     });
25
26     gpuQueue.wait_and_throw();
27
28   } catch (sycl::exception &e) {
29     /* handle SYCL exception */
30   }
31 }
```

Managing the data

Work unit

Device code

SYCL IS...

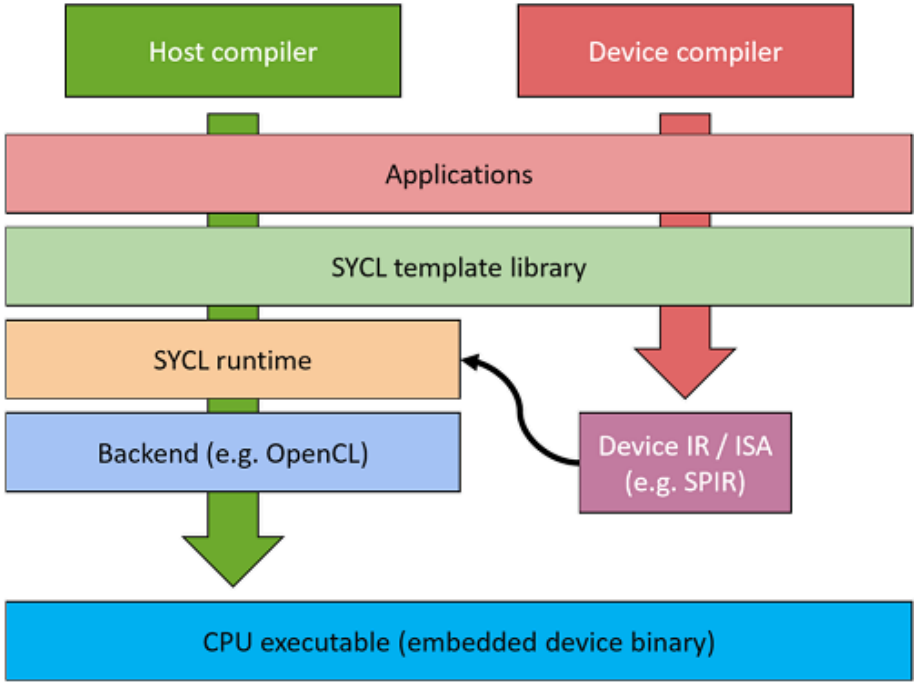
- SYCL extends C++ in two key ways:
  - device discovery (and information)
  - device control (kernels of work, memory)
- SYCL is modern C++
- SYCL is open, multivendor, multiarchitecture

FOCUS ON UNDERSTANDING THE THREE KEY CAPABILITIES:

- find devices / connect to devices
- share data with devices (get data to and from)
- offload computations to devices

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

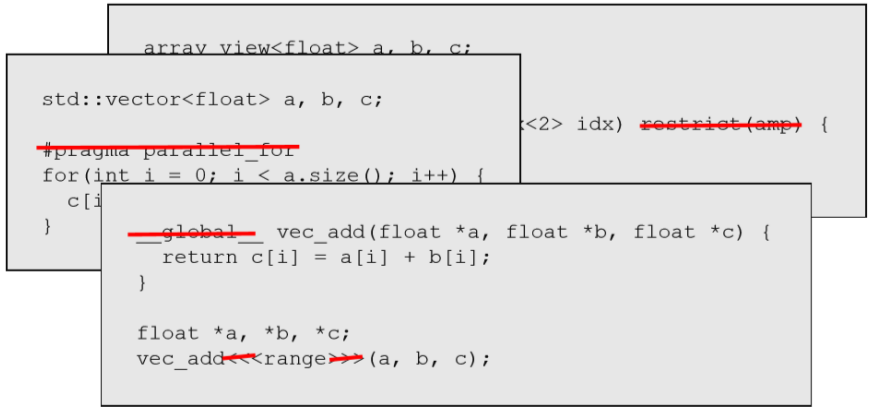
WHAT IS SYCL?

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.

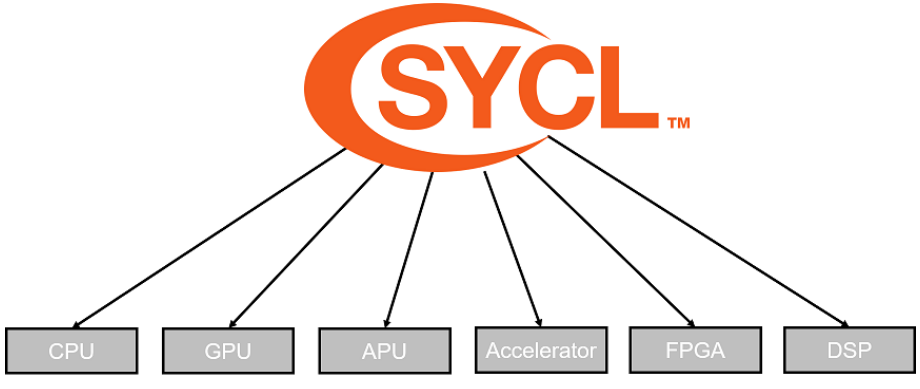



```
cgh.parallel_for(range, [=](cl::sycl::id<2> idx) {
  c[idx] = a[idx] + b[idx];
});
```

- 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?

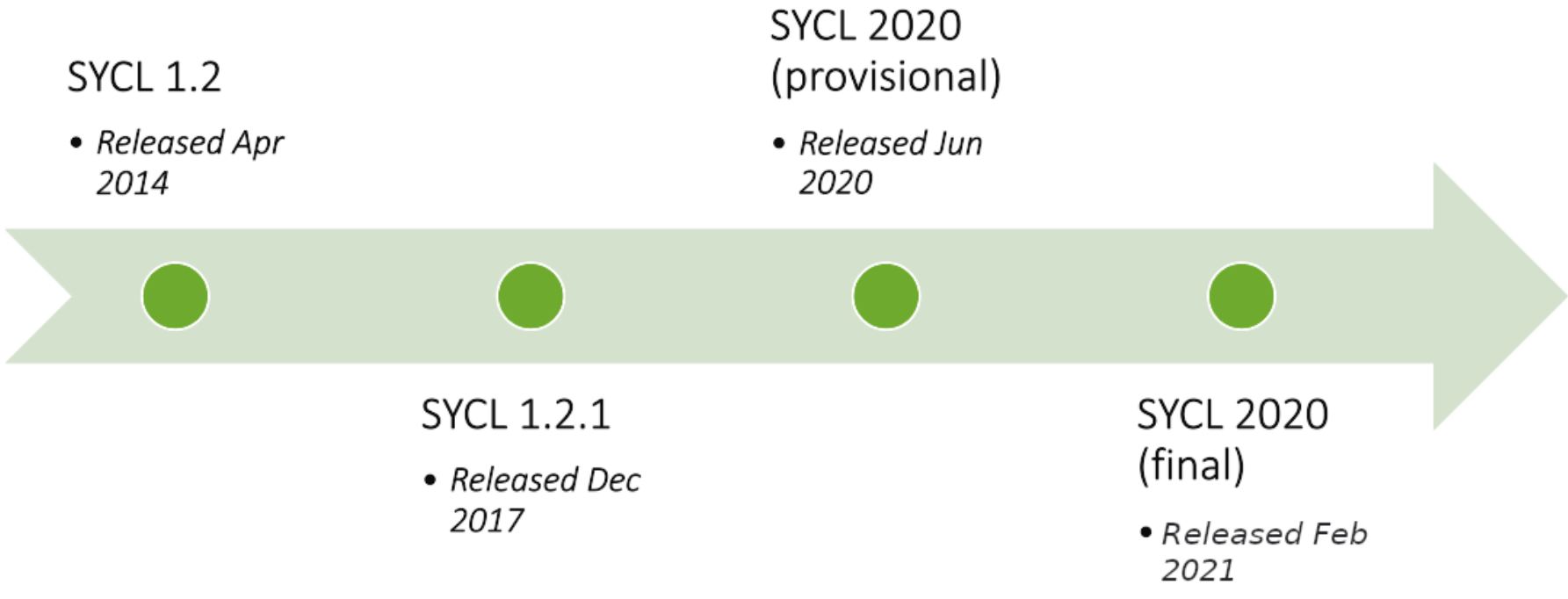
SYCL is a single source, high-level standard C++ programming model, that can **target a range of heterogeneous platforms**



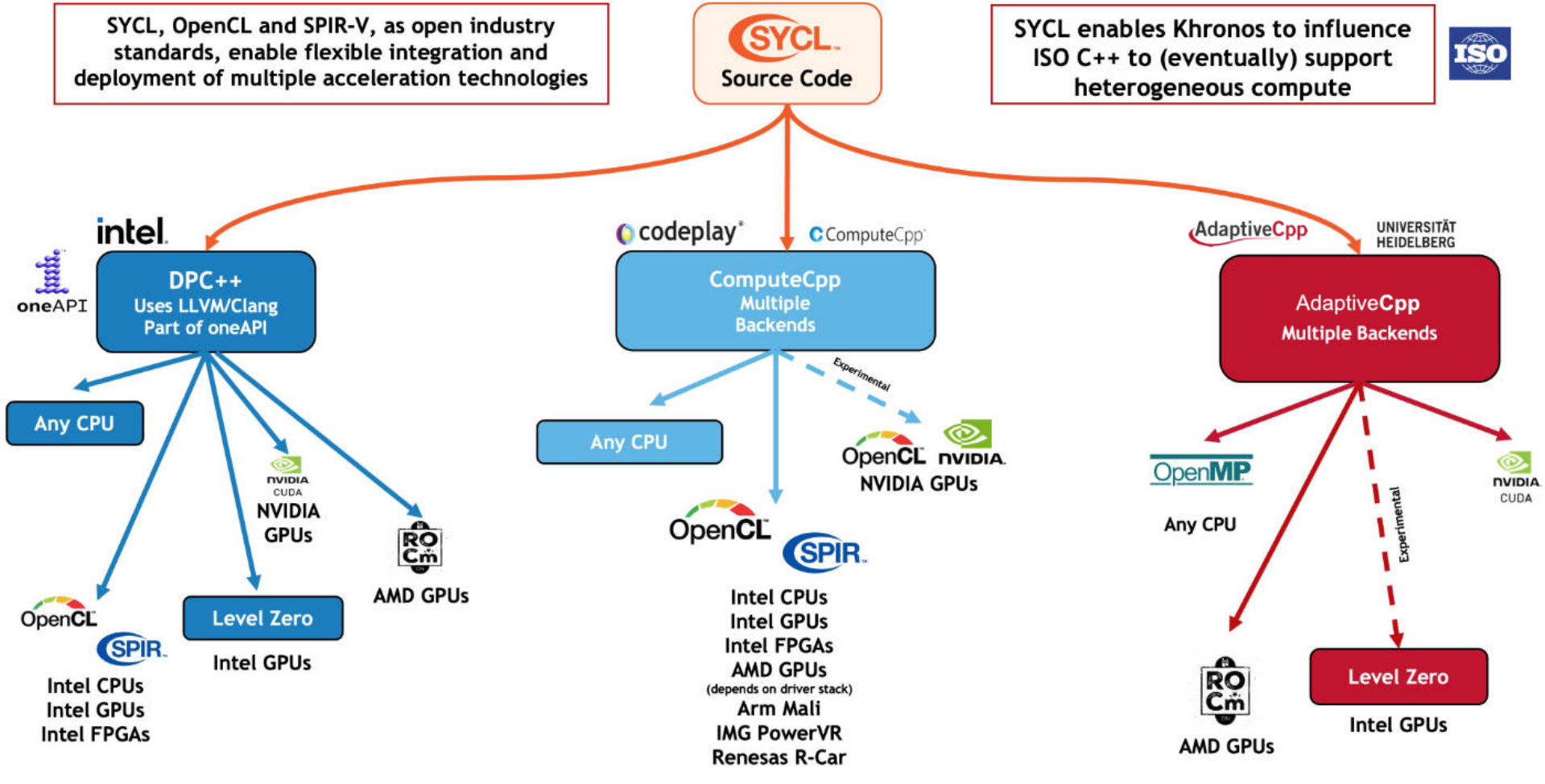
- SYCL can target any device supported by its backend
- SYCL can target a number of different backends

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.

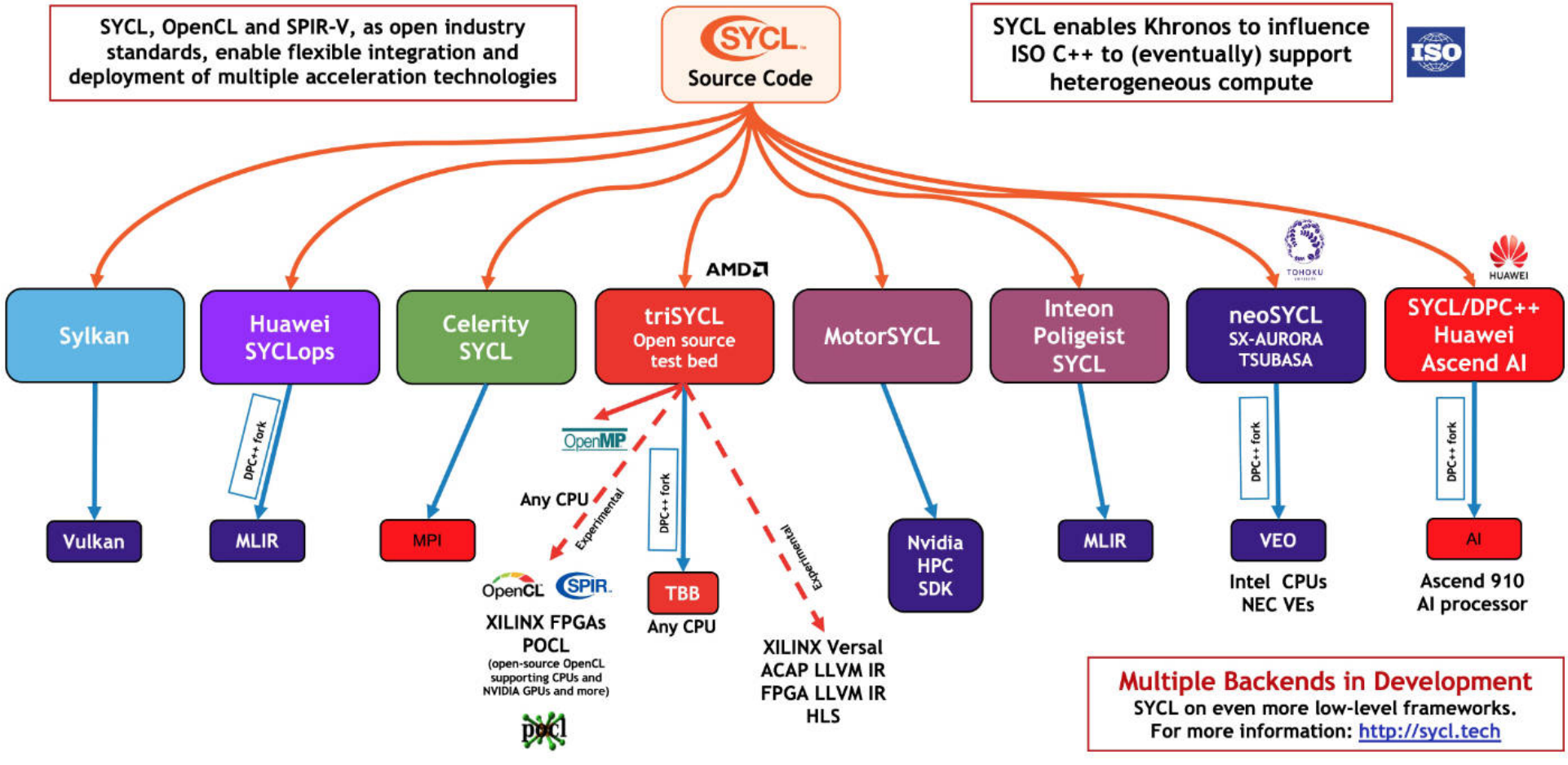
SYCL SPECIFICATION



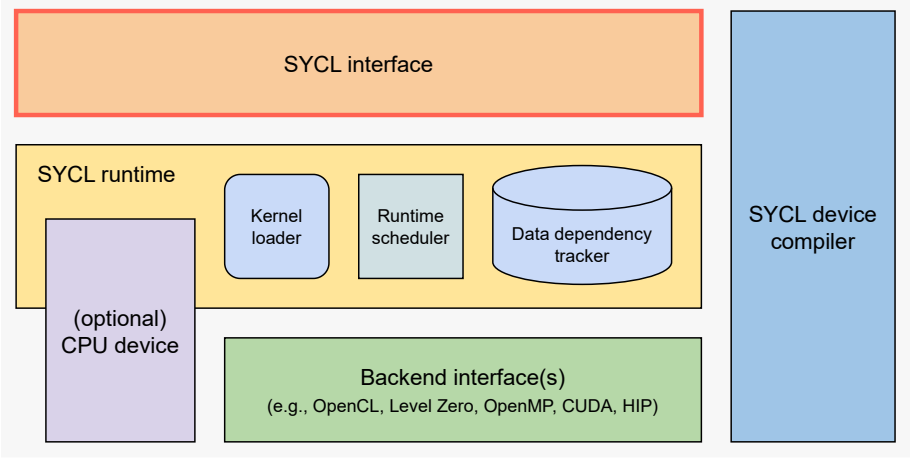
SYCL IMPLEMENTATIONS



SYCL IMPLEMENTATIONS

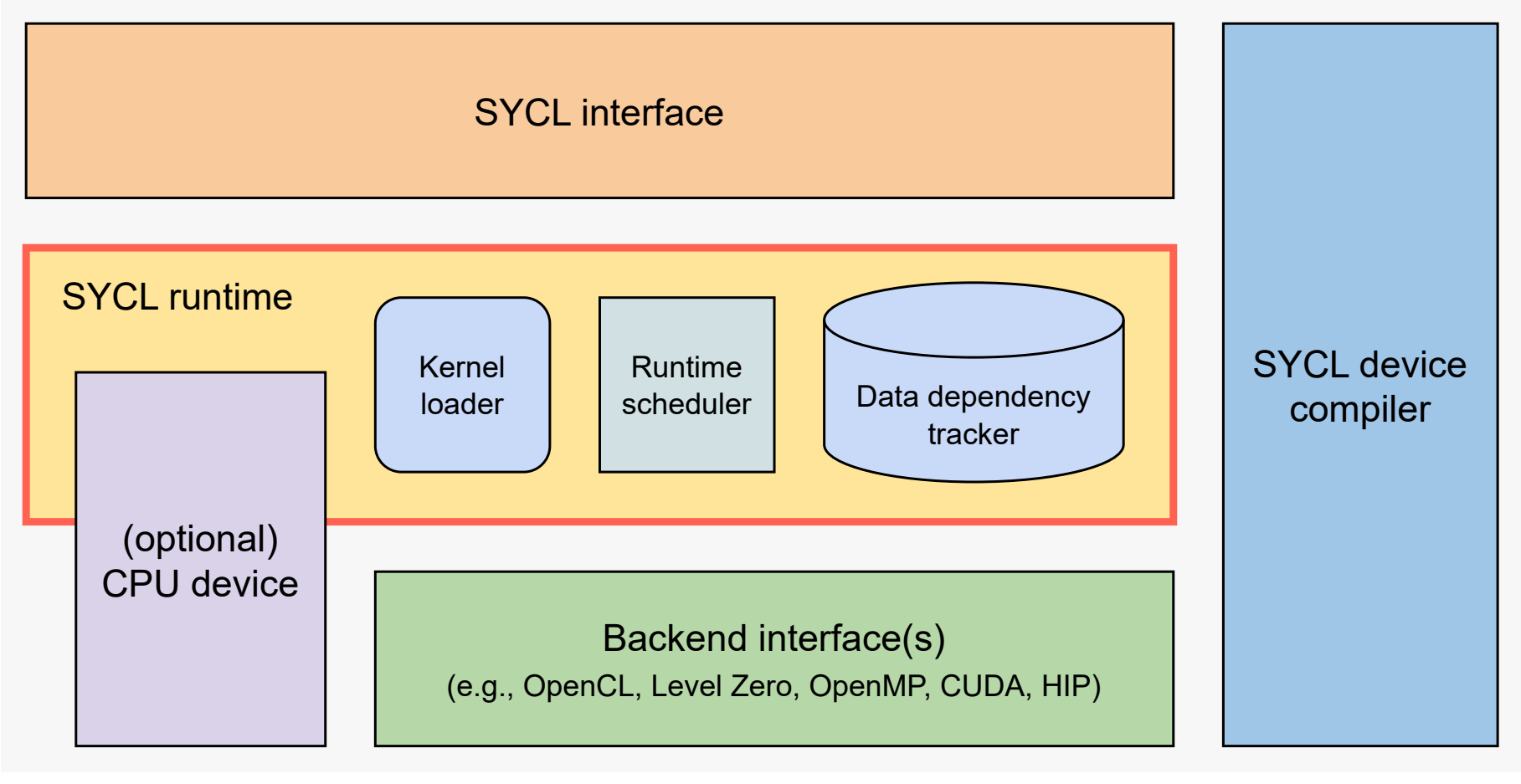


WHAT A SYCL IMPLEMENTATION LOOKS LIKE



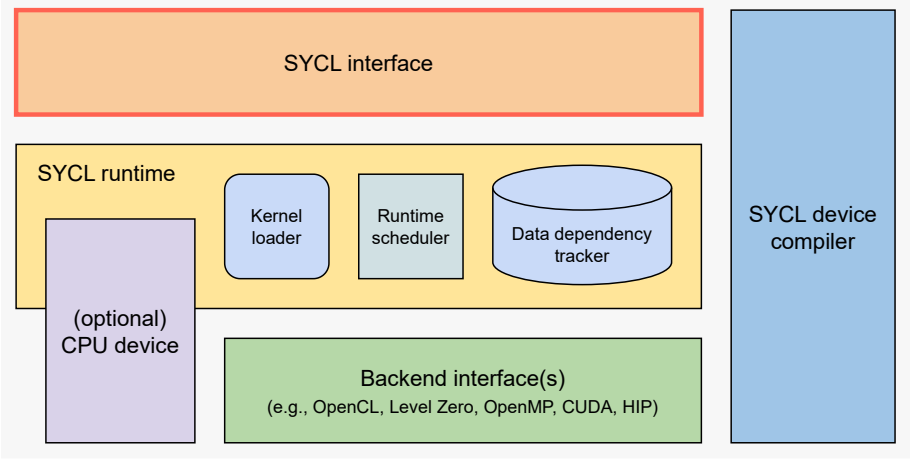
- The SYCL interface is a C++ template library that developers can use to access the features of SYCL
- The same interface is used for both the host and device code
- The host is generally the CPU and is used to dispatch the parallel execution of kernels
- The device is the parallel unit used to execute the kernels, such as a GPU

WHAT A SYCL IMPLEMENTATION LOOKS LIKE



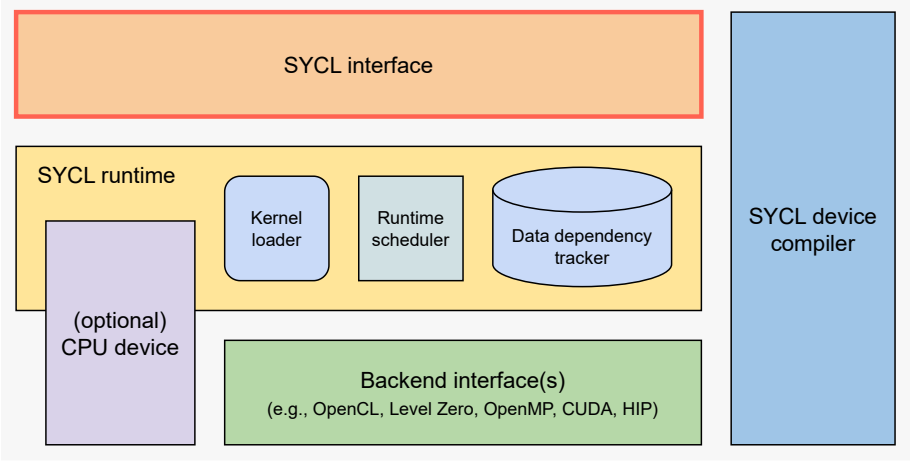
The SYCL runtime is a library that schedules, and executes work.
It loads kernels, tracks data dependencies, and schedules commands

WHAT A SYCL IMPLEMENTATION LOOKS LIKE



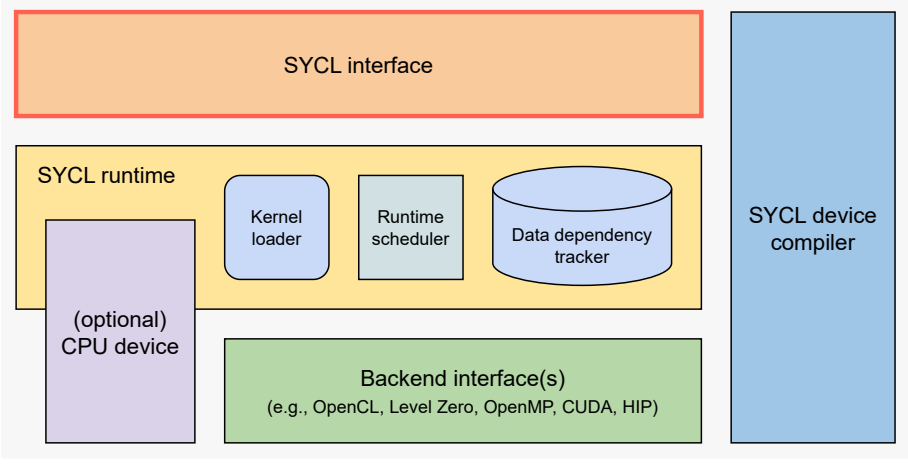
- There is no Host Device in SYCL (as of SYCL 2020)
- SYCL 1.2.1 had a concept of a 'magical' host device - an emulated backend
- SYCL 2020 implementations generally offer a CPU device
- Often, the best debugging on a platform is using a CPU device
- Yet, debugging off the CPU is important to discover offloading issues

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

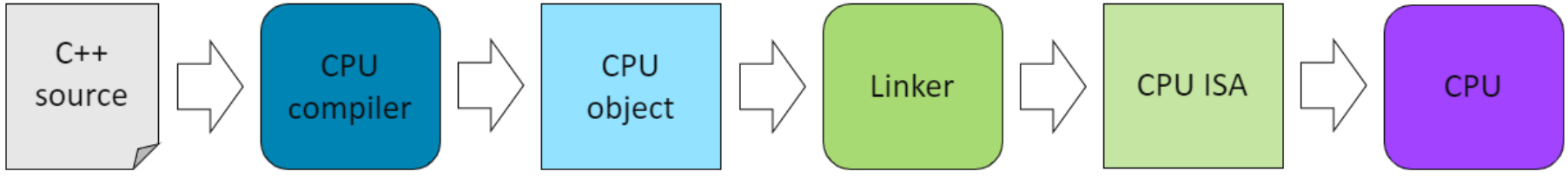


- The SYCL device compiler is a C++ compiler which can identify SYCL kernels and compile them down to an IR or ISA
  - This can be SPIR, SPIR-V, GCN, PTX or any proprietary vendor ISA

IR = Intermediate Representation

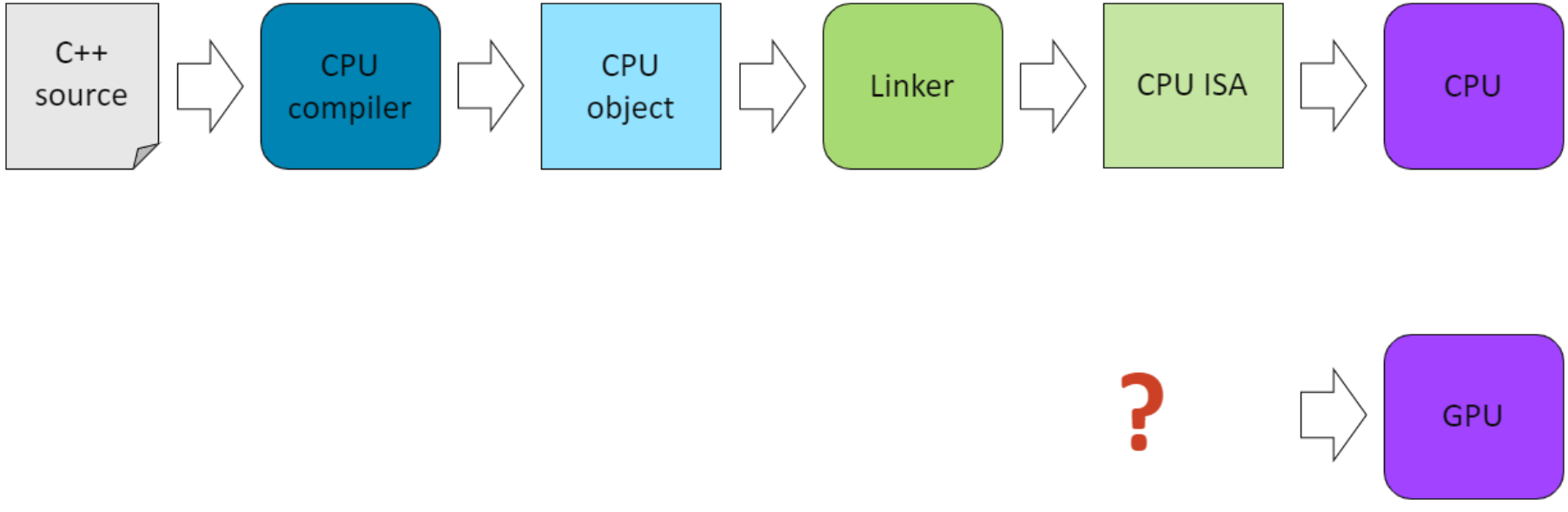
ISA = Instruction Set Architecture

STD C++ COMPILATION MODEL



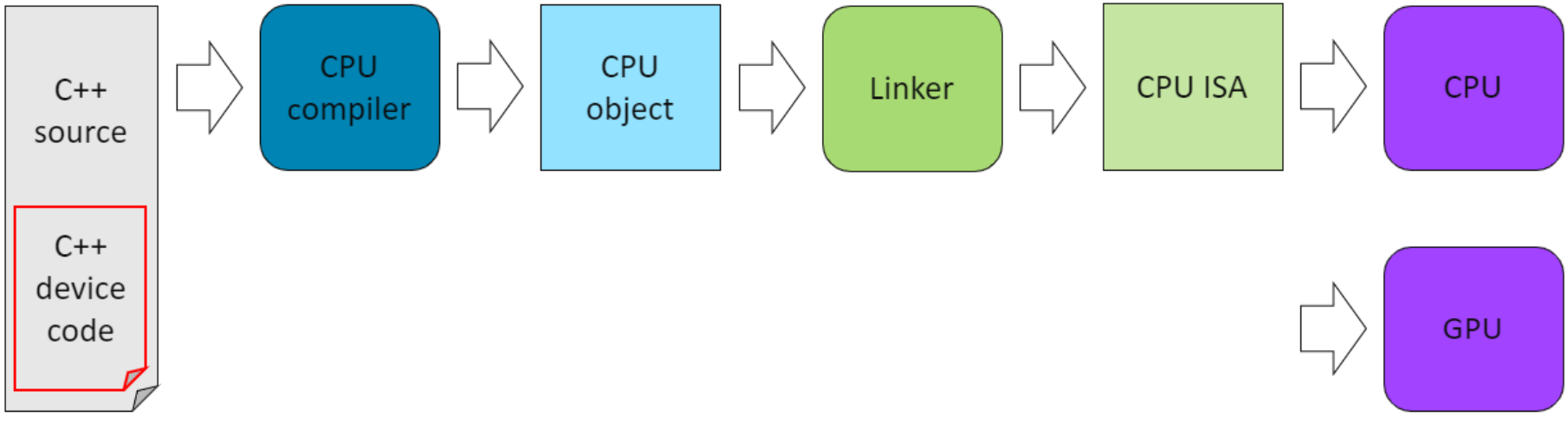
This is the typical compilation model for a C++ source file.

STD C++ COMPILATION MODEL



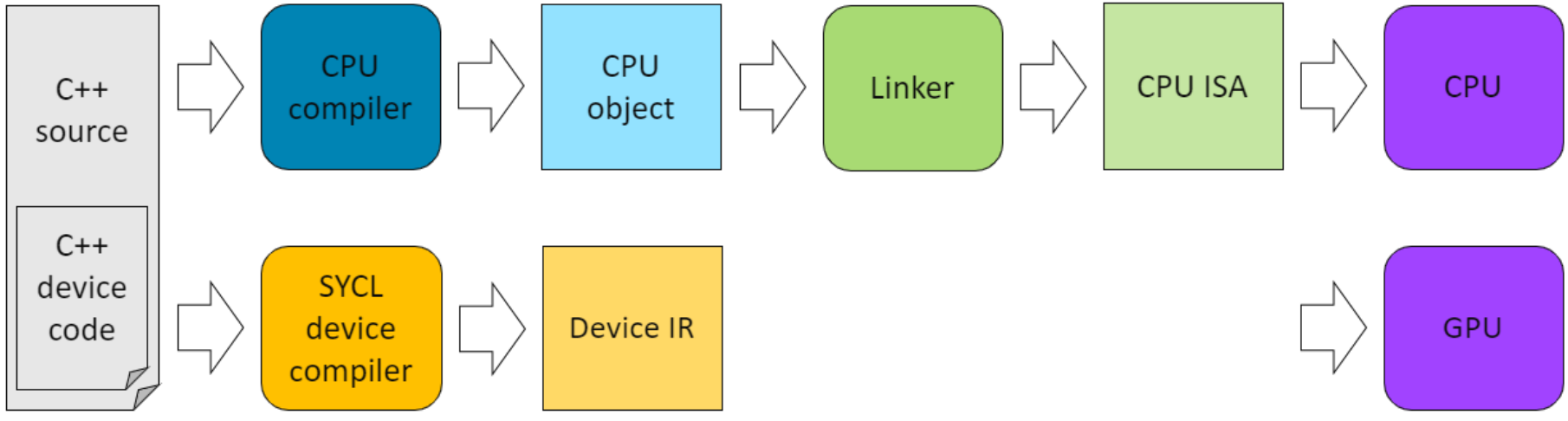
So how do you compile a source file to also target the GPU?

STD C++ COMPILATION MODEL



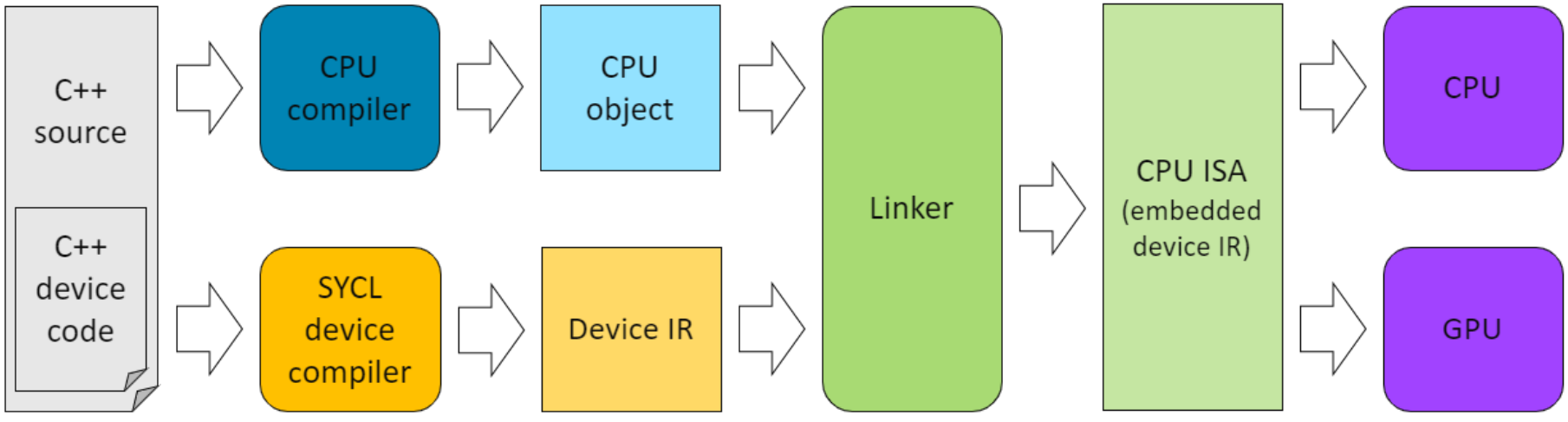
- As SYCL is single source the kernel functions are standard C++ function objects or lambda expressions.
- These are defined by submitting them to specific APIs.

STD C++ COMPILATION MODEL



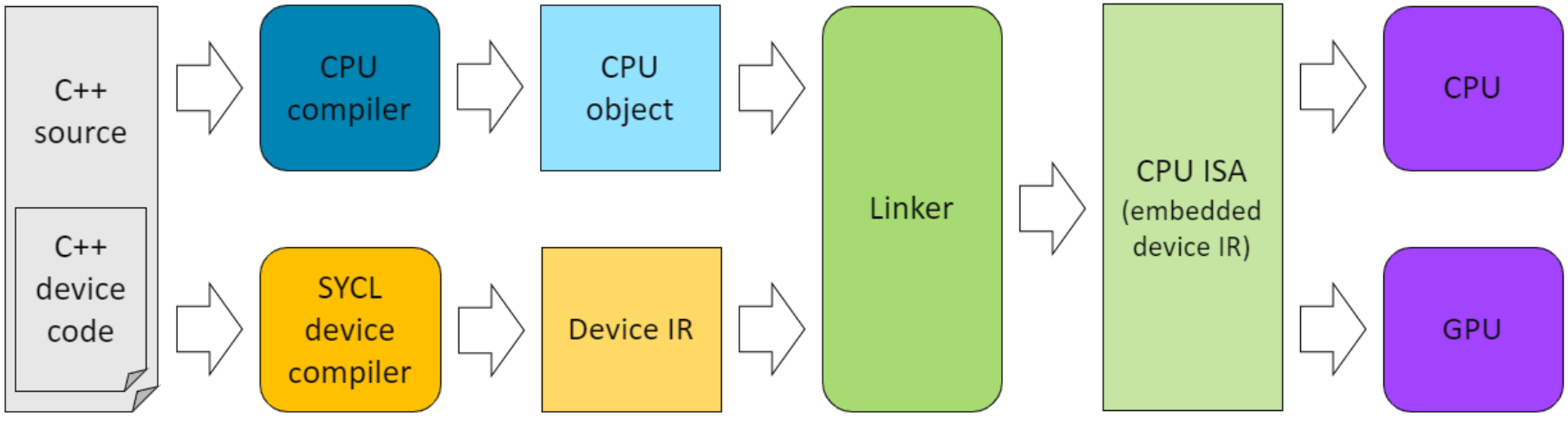
- As well as the standard C++ compiler, the source file is also compiled by a SYCL device compiler.
- This produces a device IR such as SPIR, SPIR-V or PTX or ISA for a specific architecture containing the GPU code.

STD C++ COMPILATION MODEL



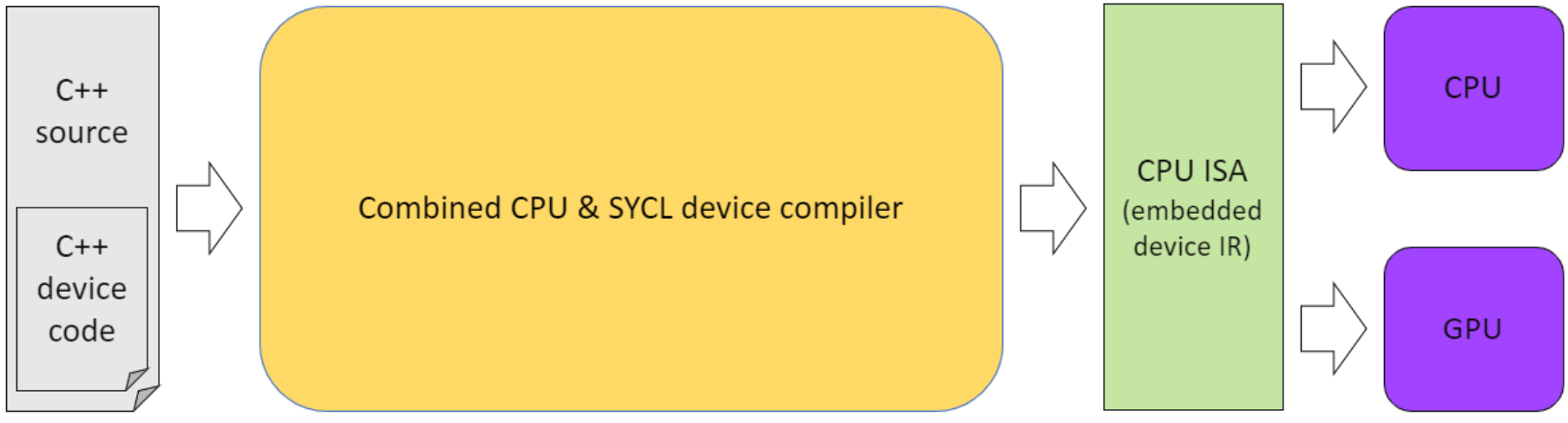
- The CPU object is then linked with the device IR or ISA to form a single executable with both the CPU and GPU code.

STD C++ COMPILATION MODEL



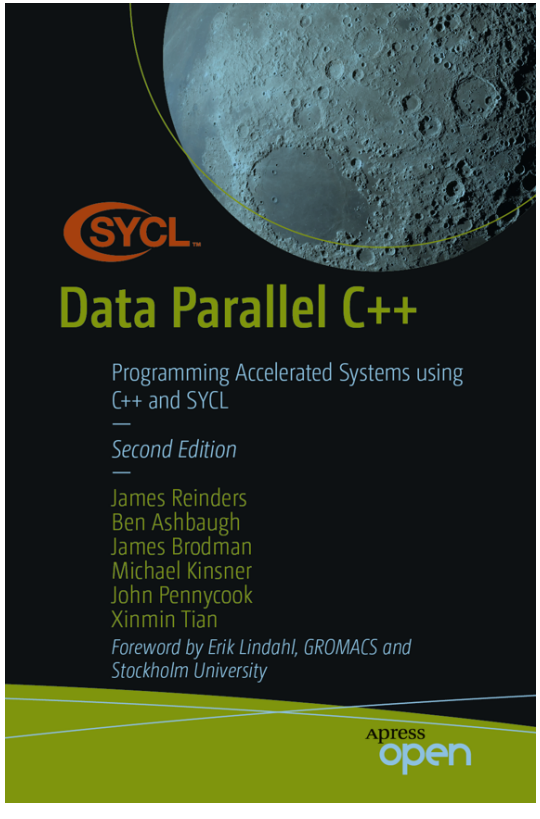
- This is the multi-compiler compilation model.
- This allows the host compiler (MSVC, clang, icx, gcc) to be independent of the SYCL device compiler.

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

QUESTIONS?

EXERCISE

Code_Exercises/Exercise_01_Big_Hello_SYCL

- Everyone set up to compile
- Everyone set up to run
- First program has a little of everything
- Focus on compile and run, and we'll grow from there

Bonus exercise: add a stream output of Hello, World!