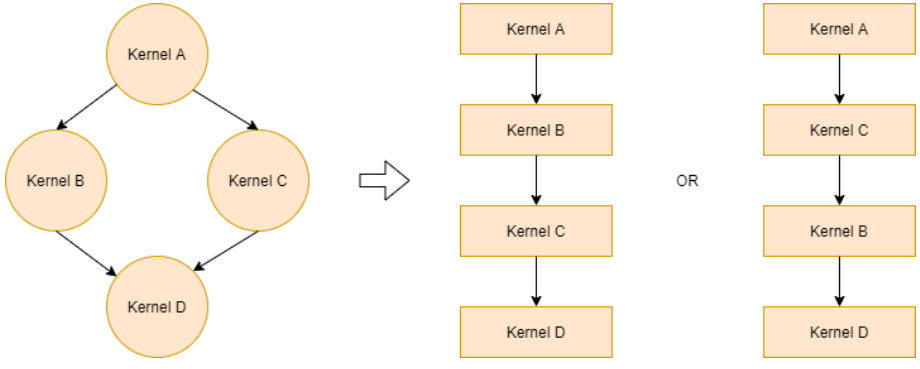
IN ORDER QUEUES

LEARNING OBJECTIVES

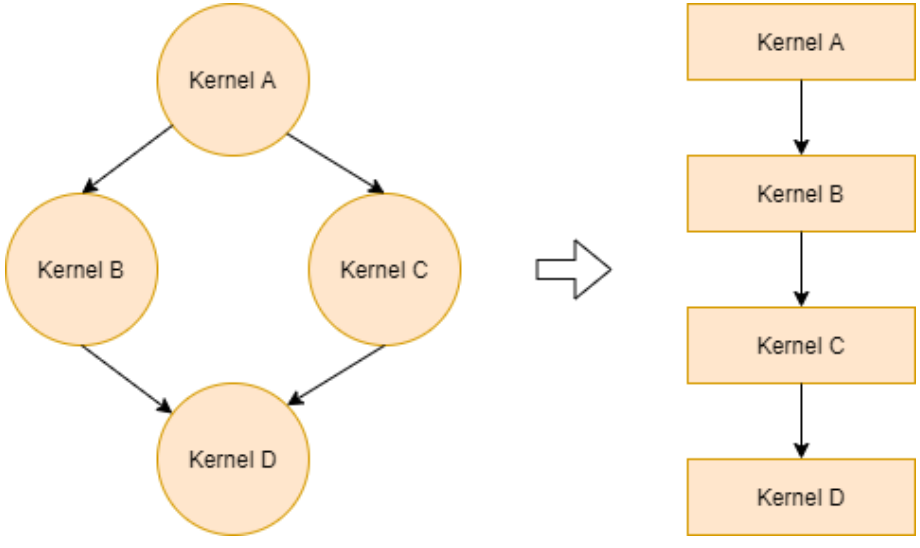
- Learn about out-of-order and in-order execution
- Learn about in-order queues and how to use them

OUT-OF-ORDER EXECUTION



- SYCL `queues` are by default out-of-order.
- This means commands are allowed to be overlapped and re-ordered or executed concurrently providing dependencies are honoured to ensure consistency.

IN-OF-ORDER EXECUTION



- SYCL `queues` can be configured to be in-order.
- This mean commands must execute strictly in the order they were enqueued.

DATA FLOW WITH BUFFERS AND ACCESSORS

```
auto inOrderQueue = sycl::queue{gpu_selector{},
  {sycl::property::queue::in_order{}}};
```

- To create an in-order `queue` simply provide the `property::queue::in_order` property to the constructor.

USING AN IN-ORDER QUEUE (BUFFER/ACCESSOR)

```
buf = sycl::buffer(data, sycl::range{1024});

inOrderQueue.submit([&](sycl::handler &cgh){
  sycl::accessor acc{buf, cgh};

  cgh.parallel_for<kernel_a>(sycl::range{1024},
    [=](sycl::id<1> idx){
    acc[idx] = /* some computation */
  });
});

inOrderQueue.submit([&](sycl::handler &cgh){
  sycl::accessor acc{buf, cgh};

  cgh.parallel_for<kernel_b>(sycl::range{1024},
    [=](sycl::id<1> idx){
    acc[idx] = /* some computation */
  });
});

inOrderQueue.wait();
```

- In the buffer/accessor model in-order queues are used as normal.
- The main difference is now that the command groups will now always be executed in the order they are enqueued.

USING AN IN-ORDER QUEUE (USM)

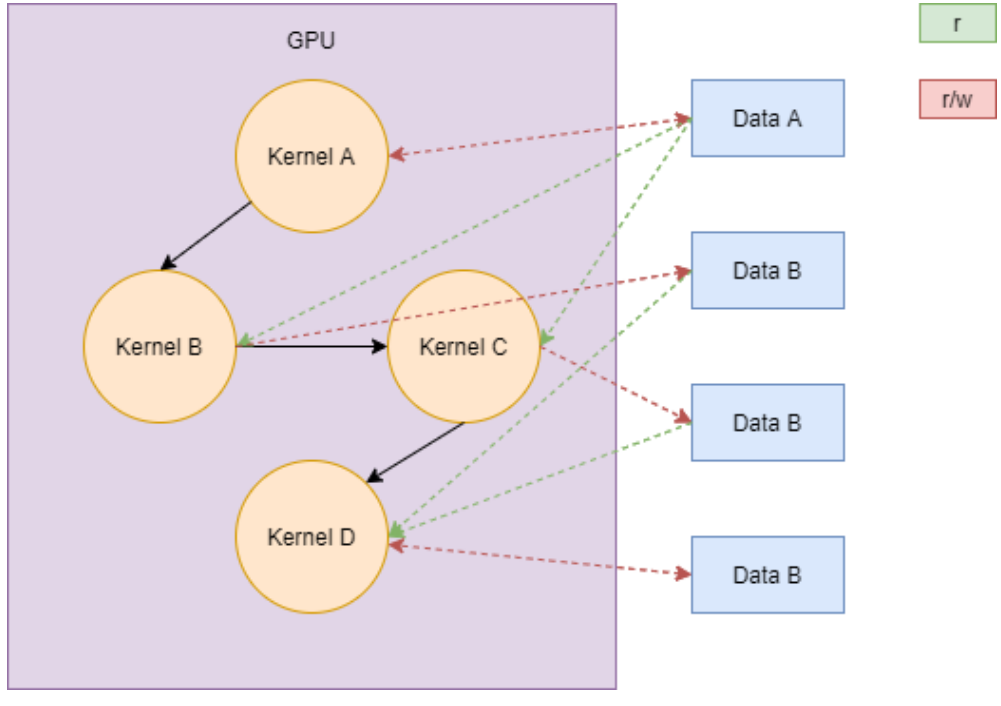
```
auto devicePtr   = usm_wrapper<int>(
  malloc_device<int>(1024, inOrderQueue));

  inOrderQueue.memcpy(devicePtr, data, sizeof(int));

  inOrderQueue.parallel_for<kernel_a>(sycl::range{10
  [=](sycl::id<1> idx){
  devicePtr[idx] = /* some computation */
});

inOrderQueue.parallel_for<kernel_b>(sycl::range{1024
  [=](sycl::id<1> idx){
  devicePtr[idx] = /* some computation */
});

inOrderQueue.memcpy(data, devicePtr, sizeof(int));

inOrderQueue.wait();
```

- In the USM model dependencies are greatly simplified.
- It's no longer necessary to chain the commands together with `event` as they execute in the order they are enqueued.
- Simply calling `wait` on the the `queue` is sufficient.

QUESTIONS

EXERCISE

Code_Exercises/Exercise_11_In_Order_Queue/source



Take the diamond data flow graph we implemented in the last exercise and convert it to use an in-order `queue`.