

# PASI 2011 - Sparse - Lab 2

## Summary

We'll go through these examples one by one in lab

### Example 1 (reduction)

```
#include <thrust/device_vector.h>
#include <thrust/reduce.h>
#include <iostream>

int main(void)
{
    thrust::device_vector<int> data(4);

    data[0] = 10;
    data[1] = 20;
    data[2] = 30;
    data[3] = 40;

    int sum = thrust::reduce(data.begin(), data.end());

    std::cout << "sum is " << sum << std::endl;

    return 0;
}
```

### Example 2 (transformation)

```
#include <thrust/device_vector.h>
#include <thrust/transform.h>

struct triple
{
    __host__ __device__
    int operator()(int x)
    {
        return 3 * x;
    }
}
```

```

};

int main(void)
{
    thrust::device_vector<int> input(4);
    input[0] = 10;
    input[1] = 20;
    input[2] = 30;
    input[3] = 40;

    thrust::device_vector<int> output(4);

    thrust::transform(input.begin(), input.end(), output.begin(), triple());

    for (int i = 0; i < output.size(); i++)
        std::cout << output[i] << std::endl;

    return 0;
}

```

### Example 3 (sorting)

```

#include <thrust/device_vector.h>
#include <thrust/sort.h>
#include <thrust/functional.h>

int main(void)
{
    thrust::device_vector<int> data(8);
    data[0] = 6;
    data[1] = 3;
    data[2] = 7;
    data[3] = 5;
    data[4] = 9;
    data[5] = 0;
    data[6] = 8;
    data[7] = 1;

    thrust::sort(data.begin(), data.end());

    std::cout << "ascending" << std::endl;
    for (int i = 0; i < data.size(); i++)
        std::cout << data[i] << std::endl;

    thrust::sort(data.begin(), data.end(), thrust::greater<int>());

    std::cout << "descending" << std::endl;
    for (int i = 0; i < data.size(); i++)

```

```

    std::cout << data[i] << std::endl;

return 0;
}

```

## Example 4 (stream compaction)

```

#include <thrust/device_vector.h>
#include <thrust/count.h>
#include <thrust/copy.h>

struct is_odd
{
    __host__ __device__
    bool operator()(int x)
    {
        return (x % 2) == 1;
    }
};

int main(void)
{
    thrust::device_vector<int> data(8);
    data[0] = 6;
    data[1] = 3;
    data[2] = 7;
    data[3] = 5;
    data[4] = 9;
    data[5] = 0;
    data[6] = 8;
    data[7] = 1;

    int N = thrust::count_if(data.begin(), data.end(), is_odd());

    std::cout << "counted " << N << " odd values" << std::endl;

    thrust::device_vector<int> odds(N);

    thrust::copy_if(data.begin(), data.end(), odds.begin(), is_odd());

    for (int i = 0; i < odds.size(); i++)
        std::cout << odds[i] << std::endl;

    return 0;
}

```

## Example 5 (host<-> copying)

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/reduce.h>
#include <iostream>

int main(void)
{
    int N = 100000;

    thrust::host_vector<int> h_data(N);
    thrust::device_vector<int> d_data(N);

    // // method 1: (one cudaMemcpy per element)
    // for (int i = 0; i < N; i++)
    //     d_data[i] = i;

    // // method 2: one cudaMemcpy for entire array
    // for (int i = 0; i < N; i++)
    //     h_data[i] = i;
    //
    // thrust::copy(h_data.begin(), h_data.end(), d_data.begin());

    return 0;
}
```